SIMULTANEOUS LOCALIZATION AND MAPPING FOR A MOBILE ROBOT OPERATING IN OUTDOOR ENVIRONMENTS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

EMRE SEZGİNALP

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
MECHANICAL ENGINEERING

DECEMBER 2007

Approval of the thesis:

SIMULTANEOUS LOCALIZATION AND MAPPING FOR A MOBILE ROBOT OPERATING IN OUTDOOR ENVIRONMENTS

Submitted by **EMRE SEZGINALP** in partial fulfillment of the requirements for the degree of **Master of Science in Mechanical Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen ______________________
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Kemal İder ______________________
Head of Department, **Mechanical Engineering**

Assist. Prof. Dr. İlhan Konukseven ______________________
Supervisor, **Mechanical Engineering Dept., METU**

Assist. Prof. Dr. Buğra Koku ______________________
Co-Supervisor, **Mechanical Engineering Dept., METU**

Examining Committee Members:

Prof. Dr. Reşit Soylu ______________________
Mechanical Engineering Dept., METU

Assist. Prof. Dr. İlhan Konukseven ______________________
Mechanical Engineering Dept., METU

Assist. Prof. Dr. Buğra Koku ______________________
Mechanical Engineering Dept., METU

Assist. Prof. Dr. Yiğit Yazıcıoğlu ______________________
Mechanical Engineering Dept., METU

Assoc. Prof. Dr. Veysel Gazi ______________________
Electric and Electronics Engineering Dept., ETU

Date: 07 / 12 / 2007

ABSTRACT

SIMULTANEOUS LOCALIZATON AND MAPPING FOR A MOBILE ROBOT OPERATING IN OUTDOOR ENVIRONMENTS

Sezginalp, Emre
M.S., Department of Mechanical Engineering
Supervisor : Assist. Prof. Dr. İlhan Konukseven
Co-Supervisor: Assist. Prof. Dr. Buğra Koku

December 2007, 75 pages

In this thesis, a method to the solution of autonomous navigation problem of a robot working in an outdoor application is sought. The robot will operate in unknown terrain where there is no a priori map present, and the robot must localize itself while simultaneously mapping the environment. This is known as Simultaneous Localization and Mapping (SLAM) problem in the literature. The SLAM problem is attempted to be solved by using the correlation between range data acquired at different poses of the robot. A robot operating outdoors will traverse unstructured terrain, therefore for localization, pitch, yaw and roll angles must also be taken into account along with the (x,y,z) coordinates of the robot. The Iterative Closest Points (ICP) algorithm is used to find this transformation between different poses of the robot and find its location. In order to collect the range data, a system composing of a laser range finder and an angular positioning system is used. During localization and mapping, odometry data is fused with range data.

Keywords: Robot Localization, 3D Mapping, Map Registration

ÖZ

DIŞ ORTAMLARDA ÇALIŞAN BİR HAREKETLİ ROBOTUN EŞ ZAMANLI KONUMLANDIRMA VE HARİTALAMASI

Sezginalp, Emre
Yüksek Lisans, Makina Mühendisliği Bölümü
Tez Yöneticisi : Yrd. Doç. Dr. İlhan Konukseven
Ortak Tez Yöneticisi: Yrd. Doç. Dr. Buğra Koku

Aralık 2007, 75 Sayfa

Bu tezde, dış ortamda çalışan bir robotun eş zamanlı konumlandırma ve haritalama problemine bir çözüm yöntemi aranmaktadır. Robot önceden bilinmeyen, haritası verilmemiş bir ortamda çalışacağından, konumlandırma sırasında etrafın haritasını da oluşturması gerekmektedir. Bu problem, literatüre Eş Zamanlı Konumlandırma ve Haritalama (EZKH) problemi olarak geçmiştir. EZKH problemi, robotun değişik konumlarda aldığı uzaklık verileri arasındaki bağlantı kullanılarak çözülmektedir. Robot, engebeli bir arazide ilerleyecektir. Bu nedenle konumlandırma için (x,y,z) koordinatlarının yanında, yunuslama, dönme ve yuvarlanma açılarının da hesaba katılması gerekir. Robotun değişik konumlar arasındaki hareketini hesaplamak ve robotu konumlandırmak için "Döngülü En Yakın Noktalar" (DEYN) algoritması kullanılmaktadır. Uzaklık verisini toplamak için bir adet lazer mesafe tarayıcı ve bir adet açısal konumlandırma sisteminden oluşan veri toplama sistemi kullanılmaktadır. Konumlandırma ve haritalama işlemleri yapılırken, odometre verisi mesafe verisi ile birleştirilmektedir.

Anahtar kelimeler: Robot konumlandırma, 3 Boyutlu Haritalama, Harita Hizalama

TABLE OF CONTENTS

To My Family

ACKNOWLEDGMENTS

I would like to thank my thesis supervisor Asst. Prof. Dr. E. İlhan Konukseven for providing me this research opportunity, and guiding me through the study.

I also would like to express my gratitude to my co-supervisor, Asst. Prof. Dr. A. Buğra Koku, not only for his guidance, but also for his friendship and encouragement in the most desperate moments during my studies.

I would like to thank my friends Anas Abidi, Faruk Yazıcıoğlu, Görkem Üner, Onur Yarkınoğlu, Süleyman Bideci and Umut Koçak, with whom I had the pleasure to study together in the METU ME Mechatronics Laboratory, for their precious support and presence. Among these precious people, I owe special thanks to Onur Yarkınoğlu, who has guided me in computer programming and allowed me to use the 3D graphical engine he has developed.

I would also like to thank my dear old friends, Burak Özkalaycı and Can Eroğul, for their patience and efforts in helping me during my studies.

Finally, I am deeply grateful to my very dear family, for their great support and love. To my family, this thesis is dedicated.

CHAPTER 1

INTRODUCTION

In recent years, there has been a significant amount of progress in mobile robot technologies, due to the vast amount of resources devoted to this area. This effort is not in vein, since mobile robots can be used in a wide variety of applications like planetary exploration, military applications, surveillance, explosive material disposal, search and rescue, hazardous material disposal, etc.

The most important goal of robotic community is making robots fully autonomous, so that no human intervention is needed for the robots to accomplish their tasks. For this goal to be reached, the navigation of the mobile robot must be fully autonomous, with no need for an a priori map of its environment or direction commands from an operator. In the past years, autonomous navigation problems for the applications in human made, indoor environments are mostly solved. However, these solutions are mostly not applicable to outdoor applications. In this dissertation, a solution method to the autonomous navigation problem in outdoor is searched.

1.1 Scope of the Thesis

This thesis is devoted to the solution of the problem of simultaneous localization and mapping for a mobile robot, in a previously unknown, unstructured environment. This is achieved by using range information of the environment, collected by a LIDAR system. The thesis can be partitioned into three main steps. First step is the design of a 3D data collection system. Second is relatively localizing the robot by using correlation between two consecutive range data taken from the environment. The last is the building of a 3D map of the environment by using the outputs of the previous step.

1.2 Outline of the Thesis

The structure of the thesis can be presented as follows:

In Chapter 2, a survey of work previously done in mapping and localization of mobile robots is presented.

The system designed for collecting 3D range data from the environment is presented in Chapter 3. The specifications and the operating principle of the system are given.

Chapter 4 presents the point matching algorithm used for finding the relative motion between different robot poses. A brief survey of point matching algorithms is presented first. Then, the applied algorithm is presented along with the filters developed for filtering the raw range data.

Chapter 5 gives the details of robot localization by presenting some results and illustrations.

Chapter 6 presents the algorithm used for diffusing the error between registrations to the global map. First, previous work in the literature is presented, then the algorithm used is given, along with some results.

Finally, Chapter 7 summarizes the current work and discusses the work done, along with the possible future work

CHAPTER 2

LITERATURE SURVEY

A mobile robot needs successful localization and a good representation of its environment to accomplish its mission. Knowledge of pose, and the locations of other places of interest in the environment are the basic foundations on which all high level navigation operations are built [1]. This knowledge enables path planning for various tasks such as goal reaching, region coverage, exploration and obstacle avoidance, and enables the robot to follow these planned paths.

2.1 Localization

Among many localization techniques, dead-reckoning is the lowest cost, most primitive and the most used form of localization technique applied in mobile robot navigation. The idea behind dead-reckoning is integration of the motion information over time, therefore, achieving a relative position estimation due to the starting position of the robot. The sources of this motion information are encoders and inertial navigation systems (INS). Encoders give the relative motion by counting the number of turns of the robot wheels, while INS's measure the angular accelerations in the three principle axis and robot pose is found by integrating these values twice. The main problem with the odometry is error accumulation. Because of the integration process, there is an accumulating error and this error becomes so large in time that the position data becomes useless for localization. There are systematic (unequal wheel diameters, wrong estimation of system parameters like wheel-base or wheel-diameters, finite encoder resolution) and non-systematic errors (travelling on uneven terrain or unexpected objects, wheel slippage) associated with the encoder data. The systematic errors accumulate constantly and can be overcome by using different techniques such as [2], [53]. However, operating in unstructured terrain, mobile robot applications in outdoor dominantly suffer from non-systematic errors. The problem with the INS is the same, since a small error in measuring the acceleration can easily cause an unboundedly growing position error in time, because of the integration process. Because of these error sources, dead-reckoning cannot be used as the only source for localization. Despite all these limitations, researchers agree that odometry is an important part of the mobile robot navigation, and is used in most of the robots, due to the following facts stated in [2]:

- Odometry data can be fused with absolute position measurements to provide better and more reliable position estimation.
- Odometry can be used in between absolute position updates with landmarks. Given a required positioning accuracy, increased accuracy in odometry allows for less frequent absolute position updates. As a result, fewer landmarks are needed for a given travel distance.
- Many mapping and landmark matching algorithms assume that robot can maintain its position well enough to allow the robot to look for landmarks in a limited area and to match features in that limited area to achieve short processing time and to improve matching correctness.
- In some cases, odometry is the only navigation information available; for example; when no external reference is available, when circumstances preclude the placing or selection of landmarks in the environment, or when another sensor subsystem fails to provide usable data.

The localization technique alternative to dead-reckoning (*Relative Localization*) is *Absolute Localization.* Localization by using active beacons, natural landmarks, GPS are examples of absolute localization.

Among these instruments, using active beacons are more suitable for indoor localization, where the environment can be engineered and beacons can be placed in most efficient locations. This technique can be used in industrial and office applications, however for localization in outdoor it is not useful. One of the main reasons is the need for placing beacons in the environment, which cannot always be performed in outdoor environments. A second reason is that the beacons must always be “seen” by the robot, however, this is not always possible due to the highly unstructured characteristics of the typical outdoor environment. For an extended survey on active beacons, one may be referred to [2].

Using *Global Positioning System* (GPS), is an absolute localization technique based on *trilateration.* The 24 satellites around earth, in groups of four, each group orbiting on planes inclined 55° with respect to earth’s equator [2]. Knowing the distance between the GPS receiver and three of the satellites enables calculating the latitude, altitude and longitude of the GPS receiver. GPS can be useful in some cases to solve the localization problem, but generally one is more interested in relative locations. For instance, knowing the relative position to a big rock or a hole is more important than knowing the exact latitude and longitude of the robot. There are some other problems associated with GPS positioning, like time synchronization between the satellites and the receiver and sufficient signal-to-noise ratio in the presence of an interference or possible jamming. Moreover, satellite signal may not be available in some applications like mine exploration applications [3], or in urban applications [4].

Landmark navigation is another localization technique used in literature. Artificial landmarks and natural landmarks are the two types of landmarks being used for landmark navigation. Artificial landmarks are easily detectable objects placed on certain positions in the environment. They suffer from the same problem with the active beacons that the environment that the robot will navigate through, must be prepared for navigation beforehand. Bearing in mind that the environment cannot always be prepared prior to mission, and the GPS data may not always be available, one may think of using the natural characteristics of the environment. For indoor navigation, structured objects like windows or door frames can be used as landmarks. However, in outdoor, extracting structured landmarks is a rather tedious task [5]. As opposed to natural landmarks in indoor environments, the ones in outdoor may not be composed of definitive geometric primitives [6]. For navigation with absolute positioning by using natural landmarks, the robot must have a representation of the environment and the positions of the landmarks. However, this a priori knowledge of the environment may not be always present.

For a mobile robot to navigate in unknown, unstructured environments, a map must be incrementally constructed, and localization must be made simultaneously while constructing the map. This problem is known as *Simultaneous Localization and Mapping* (SLAM) problem. The previous work on SLAM in the literature will be presented in Section 2.3. First, literature on the mapping methods will be reviewed.

2.2 Mapping

For successful navigation and efficient path planning, mobile robots need some kind of a representation of their working environment, that is, a map. A brief summary of mapping methods will be presented here. An extensive survey on robotic mapping techniques was conducted by Thrun and presented in [7].

Two paradigms on robotic mapping exist: *Topological Models* and *Geometrical Models.* Among these models, geometrical models give metric information like (x, y, z) coordinates of the objects, or distances of the objects to the robot, while topological models give connectivity information. Topological maps are representation of the world similar to that of the humans use when giving directions (e.g. “turn first left and go straight until you see a building”). On the contrary, distance metrics are used in geometrical models (giving an analogous example to the example above; go 100m straight, turn 90° ccw, go 200m straight). However, the difference between topological maps and geometrical maps are somewhat blur, since almost all topological representations also make use of some metric information [7].

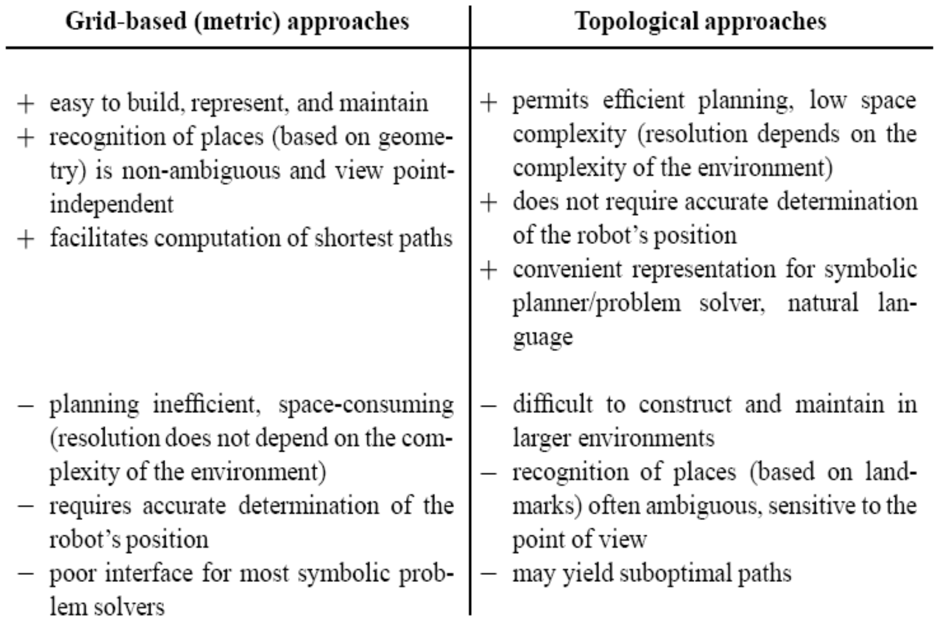
Using occupancy grids is an example of geometric mapping, where the environment is decomposed into grids, each grid, in probabilistic manner, indicating a presence of an obstacle. In reasonably small environments, as an a priori map, occupancy grids are an effective localization method. Occupancy grids offer an explicit representation of both the free space and the occupied space, which is useful for path planning. The most significant difficulty with occupancy grids when used as a priori maps in large environments, is the tradeoff between grid resolution and computational complexity. The grid size must be small enough for capturing every important detail in the environment. However, for feasible computation, larger grid size is necessary [1].

Another geometric map representation technique is, representing the environment by the global locations of distinguishable parametric features, like points, lines or circles. This map representation is called feature map. Localization is performed by extracting features from sensed data and associating these data to the features in the map. Unlike occupancy grids, feature maps form a sparse representation of the landmarks only [1]. Free space is not represented and does not take place in any of the localization process. Hence, feature maps do not facilitate path-planning or obstacle avoidance. These processes must be performed as separate operations. Another problem of feature maps is that they are suitable only to environments where the observed objects can be represented by geometric primitives. However, mostly, this is not the case in unstructured outdoor environments.

Topological maps consist of nodes and edges, and contain connectivity information. Nodes in the maps define way-points and the graph edges define procedural information for travelling between nodes. Topological maps are very useful as a priori references, since they have compact representation and logical organization for path planning. Main weakness of the topological maps is the place recognition. If the place is not recognized, or it is mistaken with another position, the graph is broken and the robot gets lost [1]. Another disadvantage of topological maps is that using purely qualitative trajectory information can work in indoor environments (like wall following [8],[9],[10]) but it is insufficient for highly unstructured environments. Moreover, robots using topological maps must use the way-points for navigation. This means constraining the robot to trajectories formed by the topological graph. This is not desirable for most of the cases. Table 2.1 gives a comparison of two mapping paradigms.

For using advantages of both mapping techniques, a hybrid approach was introduced in [11] and [12]. Topological maps are generated on top of the grid based maps, by partitioning the latter into coherent regions. This approach gains the best of both worlds: accuracy/consistency and efficiency [12].

Table 2.1: Advantages and disadvantages of the two mapping paradigms [7].



Grid-based (metric) approaches	Topological approaches
+ easy to build, represent, and maintain	+ permits efficient planning, low space complexity (resolution depends on the complexity of the environment)
+ recognition of places (based on geometry) is non-ambiguous and view point-independent	+ does not require accurate determination of the robot's position
+ facilitates computation of shortest paths	+ convenient representation for symbolic planner/problem solver, natural language
– planning inefficient, space-consuming (resolution does not depend on the complexity of the environment)	– difficult to construct and maintain in larger environments
– requires accurate determination of the robot's position	– recognition of places (based on landmarks) often ambiguous, sensitive to the point of view
– poor interface for most symbolic problem solvers	– may yield suboptimal paths

2.3 SIMULTANEOUS LOCALIZATION AND MAPPING

Mapping the environment when true pose of the robot is always known, and conversely, finding the coordinates and the orientation of the robot, given the a priori map of the environment, are solved problems in the robotic community. However, what makes a mobile robot truly autonomous, is the ability to localize itself and form a map of the environment, with unknown location in an unknown environment. This ability can be achieved by solving the well known "*Simultaneous Localization and Mapping*" (SLAM) problem (also known as Concurrent Mapping and Localization (CML) problem). The word "simultaneous" reflects the fact that the robot needs a map for localization, and for a map, the robot's true pose must be known. Thus, these two tasks must be handled in a simultaneous manner. The solution of the SLAM problem would eliminate the need for artificial infrastructures for navigation, along with the need for a priori map knowledge [13].

2.3.1 SLAM Algorithms

The SLAM problem has been approached and solved in different forms. Three main different philosophies trying to solve the SLAM problem include; probabilistic approaches, qualitative approaches and numerical approaches.

Probabilistic methods model the world such that the robot has probabilistic motion and the positions of the landmarks in the environment has uncertainties associated with them. By integrating these two distributions, and using a Bayes filter (a Kalman filter or a particle filter), the robot is localized. Mapping is considered as an extension to the localization process. The most popular approach in this category is the Kalman-filter (KF) approach due to the fact that it provides a recursive solution to the navigation problem, and using statistical models, it enables computing of consistent estimates of the uncertainties in both the robot pose and the landmark locations. In [13], existence of the solution to the SLAM problem by using an Extended Kalman Filter is proved, and an implementation of the algorithm is presented. The KF solution [14], involves a recursive update procedure that comprises prediction, observation, and update steps. Statistical models are used in this procedure to estimate the uncertainty in the robot and landmark locations, along with their intercorrelations. Although these models enable a thorough investigation into various SLAM properties, such as the convergence to a solution and the evolution of positional uncertainties, they are also the source of practical vulnerabilities. These vulnerabilities emerge from the fact that the statistical models are based on several underlying assumptions, which are not valid for every case [7]. Moreover, KF approach tries to solve the problem in state space, with states being the position of the robot and the positions of the features in the environment. However, as the robot moves through the environment and observes new landmarks, the size of the state vector grows as the number of landmarks increases, and this increases the computation time and complexity. An alternative to Kalman Filtering is the Expectation Maximization algorithms. Thrun *et.al* [15] approach the SLAM problem as a constrained, probabilistic maximum likelihood estimation problem. With this approach, they can map large-scale cyclic environments with size up to 80m by 25m. Expectation maximization (EM) algorithms have been found to generate consistent maps of large-cyclic environments, even in the presence of similar features [7]. However, EM algorithms cannot build a map incrementally since they have to process the data multiple times. The particle filtering method [16] is another important alternative to the Extended Kalman Filter. In particle filtering, continuous distributions are approximated by discrete random measures, which are composed of weighted particles, where the particles are samples of the unknown states from the state space, and the particle weights are "probability masses" computed by using Bayes theory. The basic Monte-Carlo Localization algorithm [16], [17], [18] is an example that applies particle filtering to maintain robot pose estimates. Compared with Kalman filter, particle filters have the advantage of being able to represent multi-modal distributions (that is, the robot may be in more than one place at a time). Given a sufficiently large particle set, the particle filter will always converge to the correct robot pose [27]. However, particle filtering methods for the solution of SLAM problems work well in flat and structured 2D environments but an extension to the third dimension is missing since the algorithm do not scale with additional dimensions [40].

Qualitative methods do not need absolute pose estimates as probabilistic methods do. Notable work using qualitative approaches include [8],[9]. In these works, instead of using pose estimates, the relational knowledge of the relative position of the robot and the landmarks is used.

Implementing SLAM in 3D is more difficult than implementation in 2D. In 3D navigation, the problem involves added complexity due to added degrees of freedom added to the robot motion model, and more importantly; greatly increased sensing and feature modeling complexity. This complexity arises from the absence of the manmade structured landmarks in most of the outdoor environments, where 3D SLAM is needed. Moreover, for SLAM in large, unbounded areas, the state vector unboundedly grows due to newly discovered landmarks, which makes using KF technique nearly impossible to use. One technique used by [19] and [20] is to use 2D SLAM with additional mapping capabilities in the third dimension. However, this restricts the robot motion to one plane. Another approach is direct extension of the 2D SLAM solution to 3D, however, for this technique, identifiable landmarks must be extracted from sensor data, which is not always possible in unstructured environments. A third technique is, acquiring a 3D scan of the environment at each pose of the robot, and aligning the pose estimates by correlating these scans. The previous work in 3D mapping will be presented in the following section.

2.3.2 SLAM in 3D

Thrun *et al.* [19] used two 2D laser range finders for acquiring data and building 3D maps of indoor environments. One laser scanner is mounted horizontally and one is mounted vertically. The vertically mounted laser scanner grabs a vertical scan line. This scan line is then transformed into 3D points using the current robot pose. Their approach combines ideas form incremental mapping (such as maximum likelihood and incremental map construction) with ideas of non-incremental approaches like

Markov Localization. However, their map building and localization algorithm apply only to indoor (or planar terrain) environments.

Another work on 3D scanning and mapping for SLAM of mobile robots is by Nüchter *et al* [21]. Since outdoor terrain is generally challenging by having uneven ground and unstructured, irregular shapes in the surrounding, this work takes all 6 degrees of freedom into account (x, y, z positions and the roll, pitch, yaw angles) while dealing with the SLAM problem. To test their algorithms, Nüchter *et. al* use the Kurt3D mobile robot platform, which is equipped with a 3D scanner constructed by using a 2D commercial laser scanner [21], [22], [23].

Brenneke, Wulf and Wagner also use 3D range data for the solution of the SLAM problem in outdoor terrain. Their work is original in the sense that they use 3D data for obstacle detection purposes and create a 2D occupancy map of the environment from the 3D data for the motion planning purposes [24]. The 2D map is called the “leveled range scan”. After acquiring the 3D point cloud, points are categorized as “overhang points”, ”obstacle points” and “floor points” by the obstacle detection algorithm. Overhang points are the points which are not directly connected to the ground, therefore not forming an obstacle for the robot. The obstacle points are used to form the 2D “leveled scan” so that an occupancy map in a plane is formed. The floor points are used to generate the traversable terrain in the map. Obstacle points are also used as natural landmarks for matching the different scans. Since Wulf et al. scan the environment in a continuous fashion, (i.e. not in a stop-scan-go fashion, like Nüchter et. al [25]) good synchronization between odometry, scanner and INS must be provided. Wulf *et. al* address a solution to this problem in [26].

Another work using laser scanners for 6D SLAM is by Howard, Wolf and Sukhatme [27]. They use two fixed 2D scanners mounted on a Segway RMP robot. One of the sensors is mounted horizontally, and one of them is mounted vertically, like the sensor configuration used in the work of Thrun *et. al* [19]. The work uses the sensors for mapping of urban environments. Other than the laser range finders, the robot uses GPS and IMU sensors. Their “fine localization” algorithm uses only the data from the laser scanners and the IMU. Further correction on a global scale on the pose estimate is done by using the data from GPS sensors and using Monte-Carlo Localization. Since the point cloud map formed after these processes is memory inefficient (a lot of space is used to store the data), a planar segmentation algorithm is used [28].

Pfaff and Burgard use 2 ½ dimensional elevation maps to represent the environment. Elevation maps are problematic due to the fact that overhang points are represented as obstacles in elevation maps. However, their algorithm classifies overhang points and overcome this problem [29]. They also use an ICP algorithm to match the consecutive scans. In [30], Burgard, Hahnel and Schulz represent a solution for filtering out the moving obstacles (e.g. humans) from the 3D map.

In [31], large scale, high resolution terrain models are formed by using range images obtained from one ground-based and one aerial sensor. Although this work concentrates on map building applications, the algorithm applies to localization as well. Localization may be achieved by computing the relative motion between the registered scans taken from different locations. This work does not use initial estimates of the pose changes as an initial approximation of the transform between two consecutive range image scans. Instead, a feature-based concept “spin-images” [32] is used. Despite being a feature-based registration algorithm, the algorithm does not need extraction of explicit features in the environment, as it relies on local shape signatures over the entire sensed surface.

The study presented in this dissertation tries to solve the SLAM problem in outdoor by using a method very similar to Nüchter’s [21] method. The system designed for data collection and the method followed are presented in the following chapters.

CHAPTER 3

THE SENSORY SYSTEM

In the current study, the SLAM will be done mainly by using 3D terrain data. The data is collected by a system composing of a laser range finder and an angular positioning system (Figures 3.3 and 3.4). In the following sections, some of the specifications of the sensory system will be presented.

3.1 Laser Range Finder

For collecting range data from the environment, an off-the-shelf product, SICK LMS-291-S05 [33] is used.

3.1.1 Operating Principle

The LMS-291 operate according to the time-of-flight principle. A light pulse emitted for a defined length of time is reflected off a target object and it is received back via the some path along which it was sent (Figure 3.1). A counter starts as soon as the light pulse is transmitted and stops when the signal is received back [34]. Using the counter value, the distance is calculated (The time between the transmission and reception of the signal is directly proportional to the distance between the object and the sensor). The basic operating principle for the system is initial pulse evaluation. That is, the first return pulse triggers the distance measurement, and the remaining pulses on the path are ignored. This eliminates spurious data coming from reflections. The emitted pulse is diverted by a rotating mirror in the scanner. Since the time-of-flight measurement runs at the speed of light, the rotation of the mirror for an individual pulse measurement is not relevant [33].

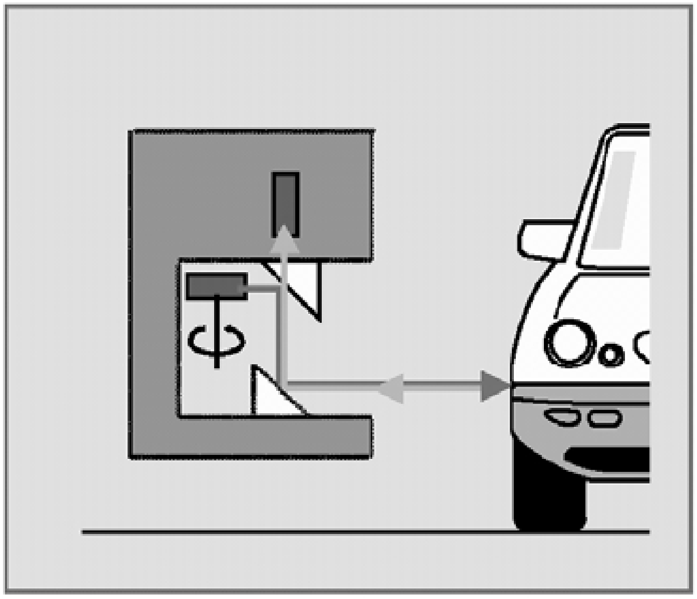


Figure 3.1 Time of flight principle for range measurement.[34]

3.1.2 Technical Specifications

The laser range finder needs 24V DC with a maximum power consumption of 20W. The power is provided by using a serial connection of two 12V-7AH Lead-Acid batteries in the current system.

The scanned area of the scanner can be selected as 100° /180°, however, for the current application, a maximum scanning angle is needed to detect as many features of the environment as possible, so 180° option is used during the data collection. There are two possibilities for the linear resolution of the scanner: 1 cm and 1 mm. Using the 1 mm distance resolution, the scanner can measure distances up to 8m, and with the 1 cm resolution, the scanner can measure distances up to 80m. For enabling the robot to map wider areas, 1 cm resolution is the appropriate choice. The angular resolution options of the scanner are 0.25°, 0.50° and 1°. However, for using 0.25° resolution, a high-speed data transfer card is needed. To capture more features in a scan, the highest resolution available without using the high-speed data transfer card, that is 0.50° angular resolution, is used.

The LMS-291 can transfer data at transmission rates of 9600, 19200, 38400 and 500000 Bd. For the latter one, a high-speed data transfer card is needed. For other specifications of the scanner, like systematic error, measurement accuracy, etc., one may referred to [33] and [34].

For the current work, 1cm linear resolution, 0.50° angular resolution and the highest data transmission rate achievable without the high-speed data transmission card, 38400Bd is used. With 500KBd transmission rate, data from one turn of the sensor mirror is received in 13ms. However, for 0.50° angular resolution, two turns of the mirror is needed. Using 38400Bd data transmission rate, the time needed to get the 361 distance values from one scan becomes approximately 339ms.

3.2 Angular Positioning System

The LMS-291 collects distance data from only the plane perpendicular to the rotation axis of its rotating mirror. However, for capturing the important features of uneven outdoor terrain, planer range data is not sufficient. For the robot to collect 3D data from the environment, an angular positioning system is designed.

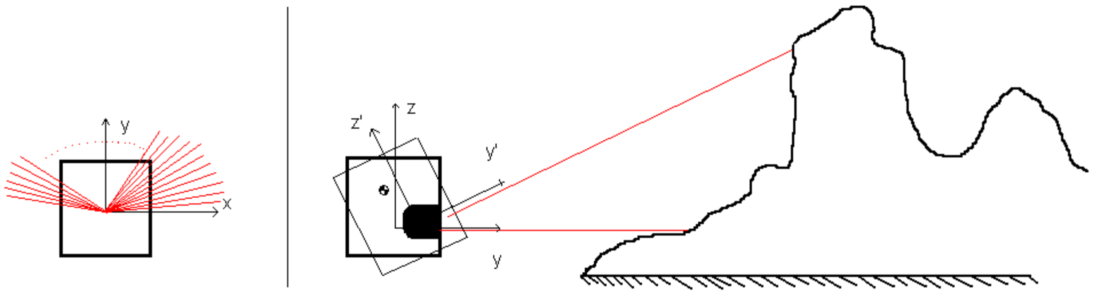


Figure 3.2 The scanner collects planar range data. To capture 3D features from the environment, scanner is given a pitching motion.

The SICK LMS-291 range finder collects range data with a 180° field of vision, restricted to a plane. Figure 3.2 (Left) shows a representation of the top view of the scanning process, while the square represents the range finder and all laser beams shown lie on the x-y plane in the figure. In order to collect 3D range data, the LMS is given a pitching motion, so that the orientation of the plane of data collection is changed (Figure 3.2, Right). The pitching motion is achieved by using a Maxon RE35 DC motor & Maxon GP32C planetary gearhead assembly. The gear ratio for the gearhead is 1526:1. For measuring the angular position of the motor, a Maxon MR digital encoder is attached to the assembly. The resolution of the encoder is 512 counts per turn. This high resolution of the encoder (2048qc per revolution), combined with the high gear ratio, allows the system to control the change of the angular position by minute angular differences. The motor has 48V nominal input voltage, however 24V input is used during the experiments due to battery weight considerations. The position control is achieved by using Maxon MIP50-E positioning controller. The batteries to supply power to the motor and the controller are two serially connected 12V-7AH Lead-Acid batteries. For communication between data collection software on the PC side and the controller, a transmission rate of 57600Bd is used.

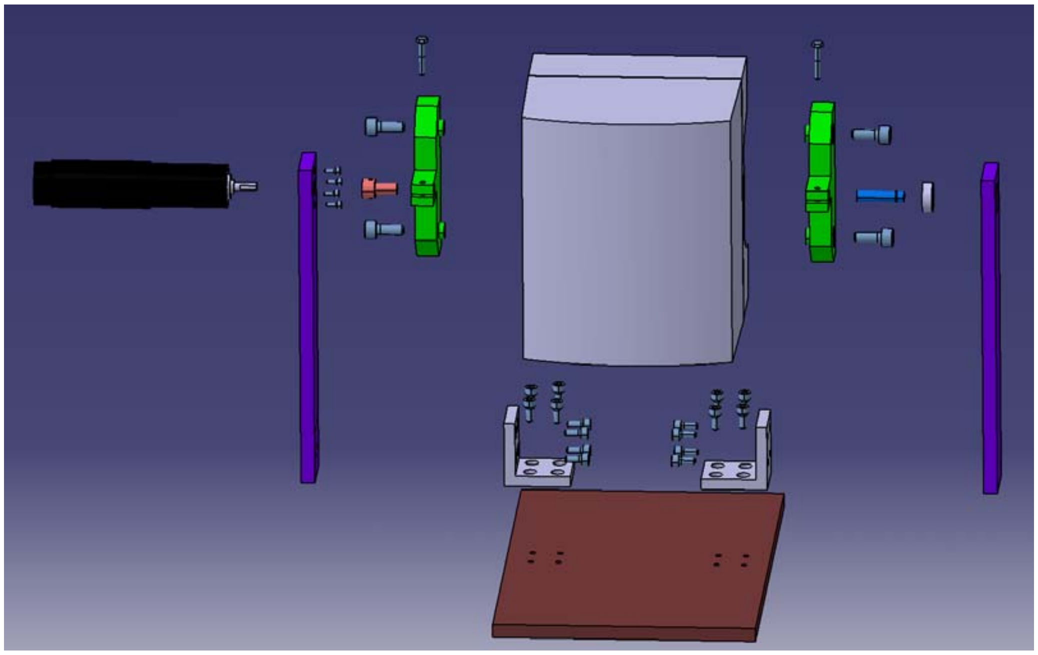


Figure 3.3 The assembly for the angular positioning system.

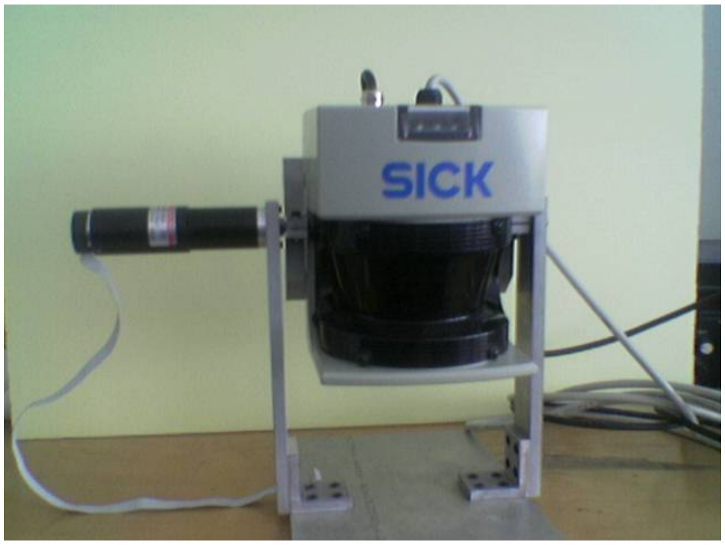


Figure 3.4 The data collection system

The vertical scanning angle for the resulting system is 80°. This is achieved by incrementing the motor position 100 times by intervals of 0.8°. Thus, we get a 3D view of 180° (horizontal)x 80° (vertical) representing the environment of the robot. As explained in the previous section, one planar scan is received in approximately 339ms. Thus, a complete 3D scan consisting of 101 such scan planes is collected in approximately 34s.

CHAPTER 4

RANGE IMAGE REGISTRATION

In order to localize the robot and map form a 3D map of the environment, the correlation between consecutive scans will be used. This technique, also used in [3], [31], [19], [23] is based on the idea of Besl [35], Zhang [36] and Chen and Mendioni [37].

4.1 Iterative Closest Point Algorithm

The solution to the optimization problem involving registration of 3D free form shapes was independently addressed by Chen and Mendioni [37] in 1991, Besl and McKay [35] in 1992 and Zhang [36] in 1994. having some differences in their algorithms, they all used a similar idea for registration. The idea is simple: Given that the motion between two successive frames (for the purpose of this thesis, scans) is "small", the geometric feature (surface, curve or point) representing the shape on the first scan is close to the one belonging to the second scan. So, assuming that the closest features (from here on the word "points" will be used instead of "features") correspond to each other, these points are matched. After the matching process, the motion that brings the two scans closer is computed by using one of the methods discussed in [38]. A fine transformation is obtained by iteratively applying this procedure to corresponding scan pairs.

Despite the fact that they use the same idea, the three works concentrating on this problem has some differences. For example, Chen's work finds the minimum distance between the point on the data set and the tangent planes on the model set. This value is then used for the point matching step. However, in Besl's work, point-to-point distance is used, whereas in Zhang's work, two constraints are used for point matching. The first of these constraints is point-to-point distance, and the second one is the angles between tangent directions. Moreover, Besl uses all the points (or features) in the data and model set , whereas Zhang applies a statistical elimination process for the points to be matched. This makes Zhang's method more robust, since Besl assumes that data set is a subset of the model set.

What ICP algorithm does is to find the optimum transformation (R,t) such that the following cost function is minimized [35], [36]:

$$E(R,t) = \sum_{i=1}^{N_P} \sum_{j=1}^{N_Q} \left\| \omega_{i,j} ((R \cdot p_i + t) - q_j) \right\|^2 \qquad \textbf{(4.1.1)}$$

Where p_i is a point on the data set, q_j is a point on the model set and $\omega_{i,j} = 1$ if p_i and q_j are corresponding points and $\omega_{i,j} = 0$ if they are not. The transformation (R,t), consisting of the rotation matrix R and the translation t, is the rigid transformation that transforms representation of the data set P (consisting of N_P points) into model set Q (consisting of N_Q points). An overview of the procedure that the ICP algorithm follows is as follows:

1. Compute the closest points on the model set those correspond to that of the data set.
2. Compute the registration, i.e., compute the transformation (R,t) (a thorough discussion on the most common methods used in computing this transformation by using the matching points can be found in [38]).

3. Apply the registration to the data set.
4. Repeat the steps above until the termination condition occurs.

The last step is applied in different ways in the different works mentioned above. Besl uses a threshold for the least squares error, $E(R,t)$ and terminates the iteration when the error falls below this preset threshold. Zhang uses the change in the motion estimate between two successive iterations. The proof of convergence of the ICP algorithm to a minimum can be found in [35].

4.2 Current Study

In this thesis, an approach very similar to Zhang's approach [36] will be used while matching two consecutive scans. As stated in Chapter 3, each 3D scan consists of 101 planar scans consisting of 361 data points taken at different angles to the horizontal plane. The matching process can be done by matching the whole 3D point data set or by extracting a horizontal plane of fixed height from both scans and matching these two horizontal planes and then applying the registration to whole 3D point set [3]. The latter one can be useful if the robot moves on a planar path, since it decreases the computational cost. However, it cannot be applied to a robot moving in non-planar terrain. Moreover, using the whole 3D data set has the advantage of reflecting a larger set of attributes [3]. The following sections present the procedure used for scan registration.

4.2.1 The Scan Matching Algorithm

The algorithm used can be summarized as follows:

1. A preprocessing step is applied to the two consecutive scans (Filtering, data reducing, computing the normals at each data point).
2. A kD-tree structure of the second scan for fast nearest-neighbor searching is formed.

3. The initial transformation guess coming from the odometry data is applied to all the points in the first scan.
4. For sampled data points in the first scan, the corresponding point in the second scan is found.
5. Corresponding point pairs are updated according to the statistical approach given in [36].
6. A rigid transformation is obtained by using the corresponding point pair set obtained in the previous step.
7. The rigid transformation found in step 6 is applied to all the points in the first scan.
8. Steps 3 to 7 are repeated until the change in error $E(R,t)$ between consecutive iteration steps is below the preset threshold.

4.2.2 Preprocessing of the Data

Since the scans will contain some erroneous points, using all the points in consecutive 3D scans in their raw form for the matching process would result in erroneous registration of the scans.

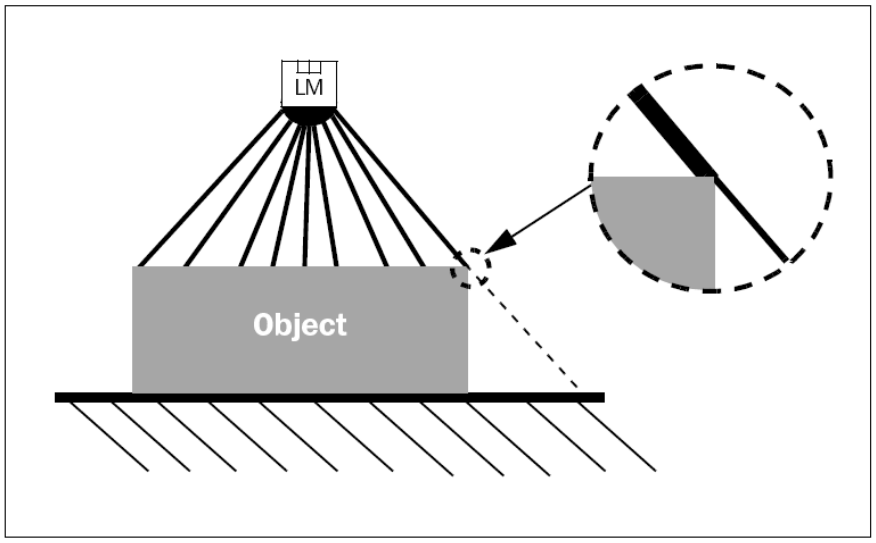


Figure 4.1 The “edge strike” [34]

The erroneous points may occur due to three reasons. The first reason is simply Gaussian noise due to the sensor characteristics. The second one is the "salt and pepper" noise, which can occur when the laser spot hits just the edge of an obstacle (Figure 4.1) . When an edge strike occurs, the measured distance using the beam falling on the edge is a combination of the foreground object and the background object, i.e. the range measurement falls in between the background and foreground objects [39]. An example is shown in Figure 4.2.

The third erroneous data source is that when an area is scanned, if there is no obstacle within the range of the scanner for a certain area, the data collection program gives the information that there is an object at distance 81.89m at the corresponding point. These spurious points must be also eliminated before proceeding to scan matching.

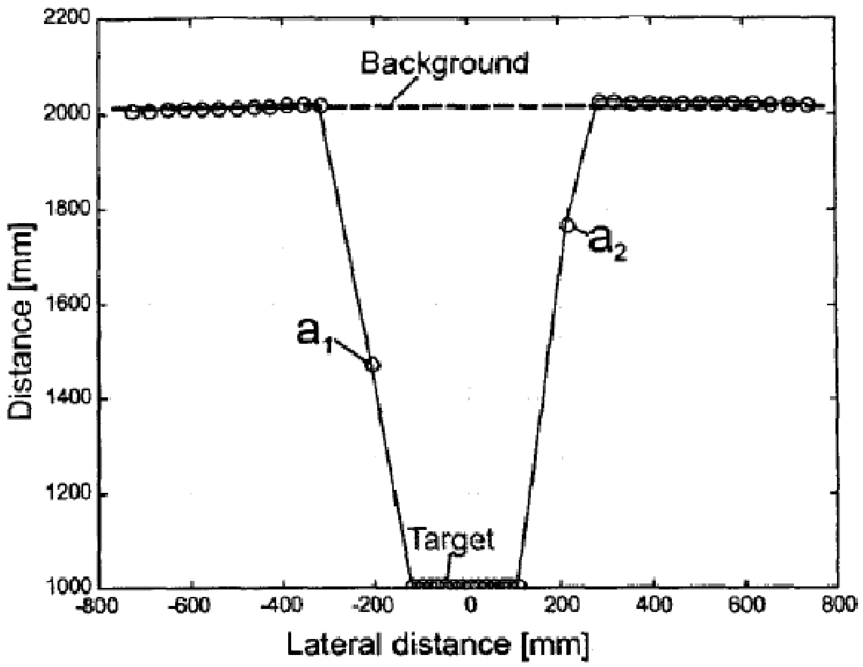


Figure 4.2 The spurious points a_1 and a_2 occur due to the edge strike [39]

In this study, the “salt and pepper” noise is eliminated by using a simple median filter. For lowering the Gaussian noise, a point reduction algorithm presented in [3] is used. After filtering the outlier points and reducing the data, the spurious points at 81.89m distance to the origin of the scan are deleted from the point cloud.

4.2.2.1 Median Filter

Each scan must be smoothed by getting rid of the outlier points. This is done by using a simple median filter, very similar to the filter used in [3] and [40]. In these studies mentioned, the median filter uses the fact that data from the scanner outputs the measured points consecutively, that is, first data point is taken at 0°, the second is at 0.5°, and so on. The median filter takes a planar scan, and removes a point if and only if the difference between distance of the point to the origin and the median value of distances of surrounding points is larger than some threshold. The surrounding points are found by using their place within the scan. The value of the distance of the removed point is changed with the median of its 7 surrounding points [3]. The effect of the filter can be observed in Figures 4.3 , 4.4 and 4.5

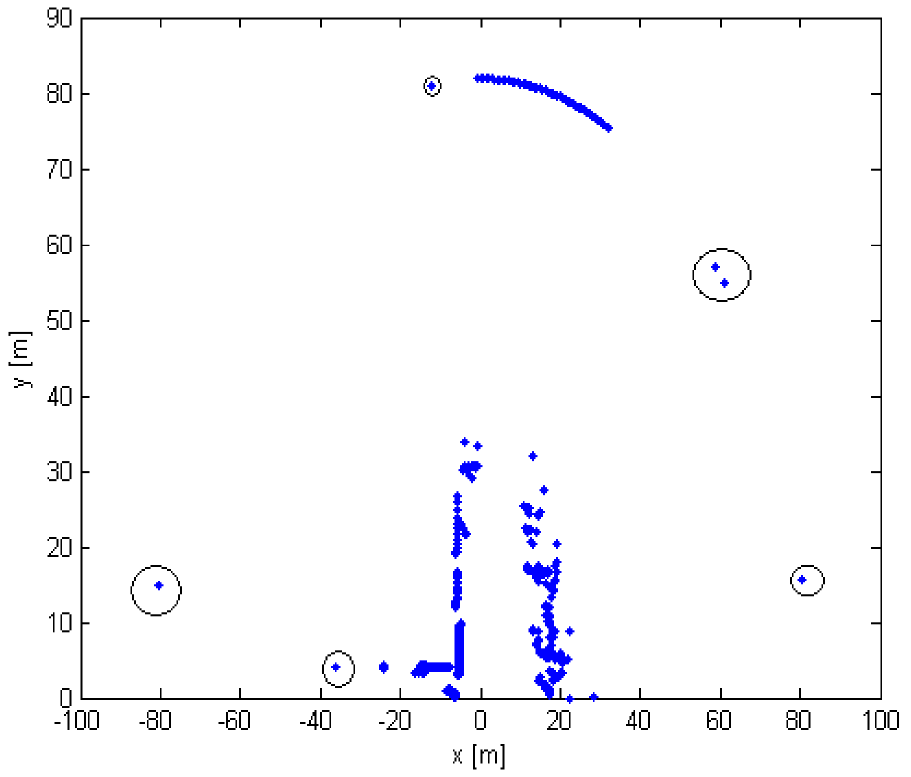


Figure 4.3 The “salt and pepper” noise. Noisy points are marked with black circles.

The marked big arc remaining on the top side of Figure 4.4 is composed of points with maximum value (d=81.89m) returned from the sensor. This indicates that there are no obstacles ahead. These points can be deleted after point reducing. Connecting the dots and representing both scans in the same graph may give a better idea of the benefit of the median filter (Figure 4.5).

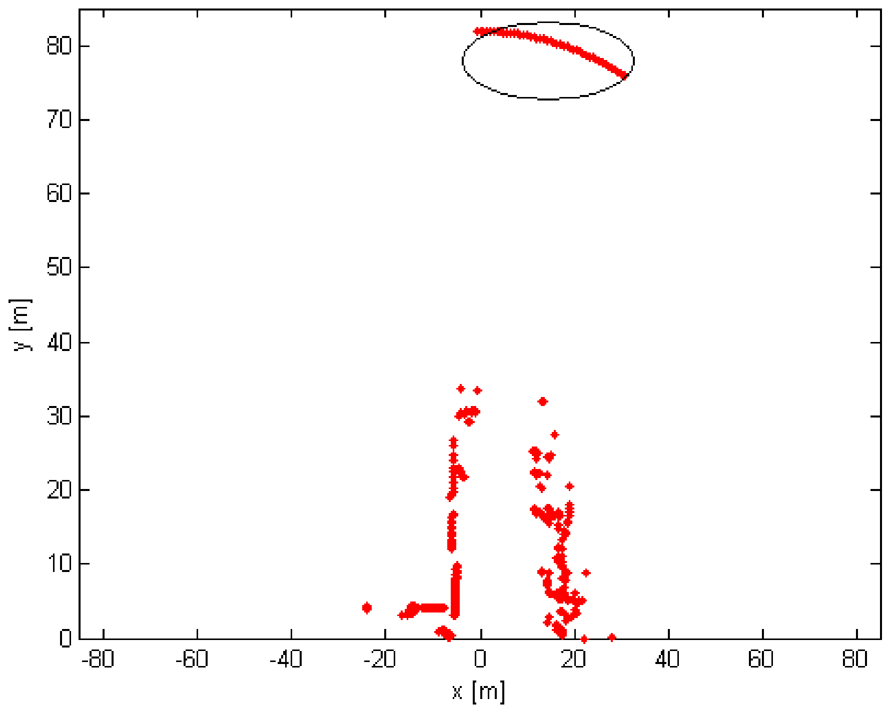


Figure 4.4 The erroneous points in Figure 4.3 are filtered.

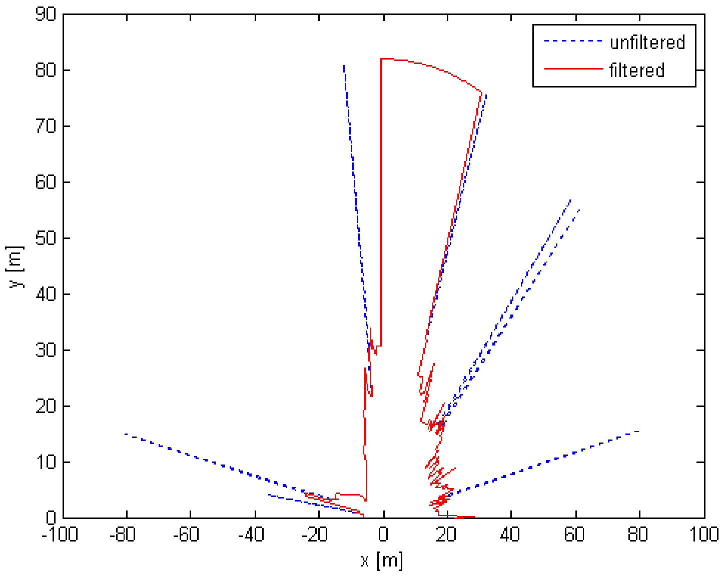


Figure 4.5 The difference between the scan filtered by the method in [3] and unfiltered scan.

In this study, instead of using the distance values of neighboring points in a planar scan like Thrun et al. [3], neighboring points in adjacent planar scans are also used while applying the median filter. Taking distance measurements coming from the scanner as a 101 x 361 dot matrix, a 3x3 window is moved around the matrix and the center point of the window is filtered by using the median of the distance values in this 3x3 window. That is, if the center point's distance value is larger than the median of the distances in the window by a fixed threshold, the distance value of the center point is changed with the median value.

Using filtered data obtained by using the method in [3] resulted in more erroneous results during pairwise scan registration than the above mentioned method . This is due to the fact that using only neighbors in the planar scans during median filtering resulted in some false-positives, and the method of moving a 3x3 window around the dot matrix of distances resulted better rejection of the outlier points most of the time.

The data in Figure 4.6 was obtained in the METU Mechanical Engineering parking area. Figure 4.6 represents a full scan acquired at a single point, with gray scale values representing the depth values where pure white pixels represents data at distance d>80m and pure black pixels represent data at d=0m. The threshold for the median filter was set as 2m. That is, if the difference of the distance of the point in question and the median of the distances in its eight-neighborhood is more than 2m, the value of the distance of the point is changed and set to the median value. One can see that outliers in the marked area in Figure 4.6 are smoothed better if points in the adjacent planar scans are also used. The data in Figure 4.7 was taken after a straight ahead movement of approximately 8m. Formation of the false positives when a median filter is applied to planar scans separately is observed in the right side of the figure.

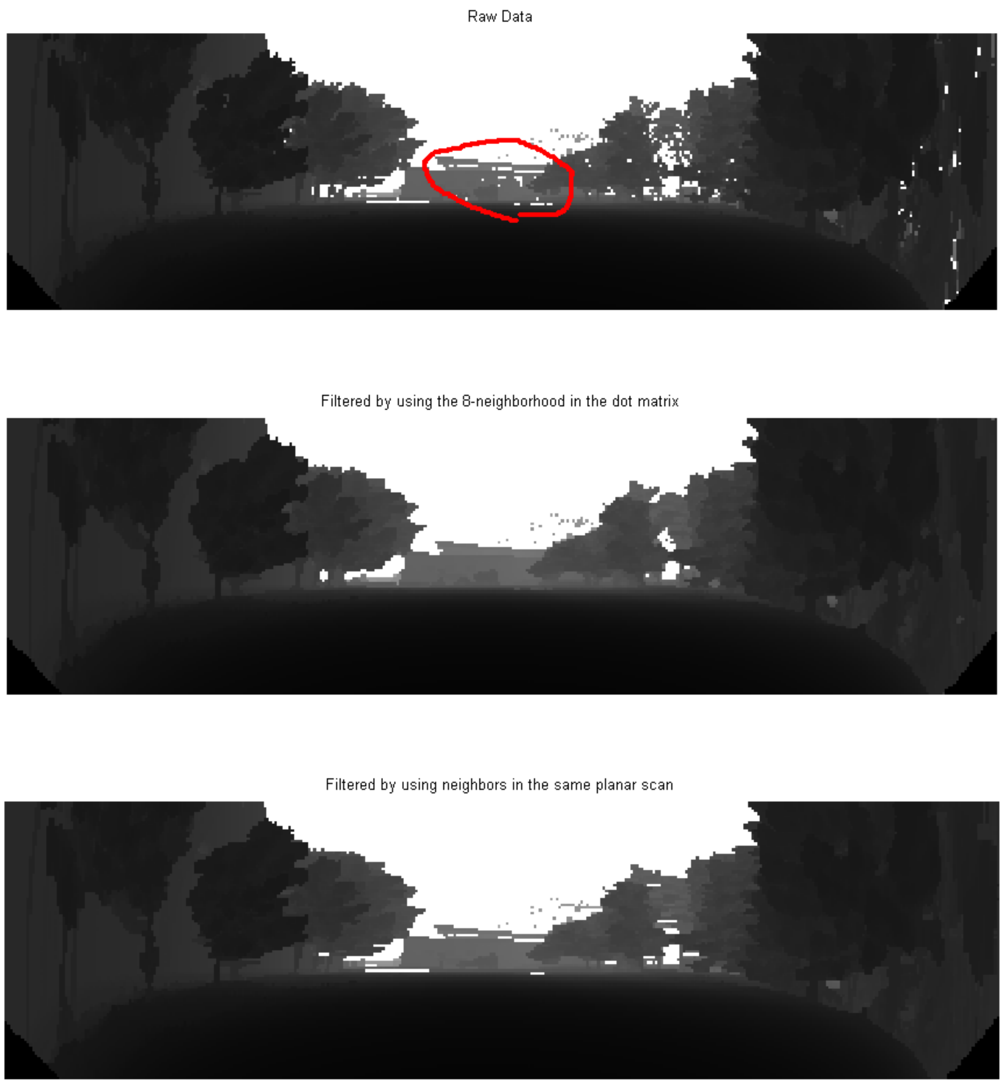

Raw Data
Filtered by using the 8-neighborhood in the dot matrix
Filtered by using neighbors in the same planar scan

Figure 4.6 Top: Raw Data, Middle: The current filter is applied Bottom: Filter in **[**3**]** is applied.

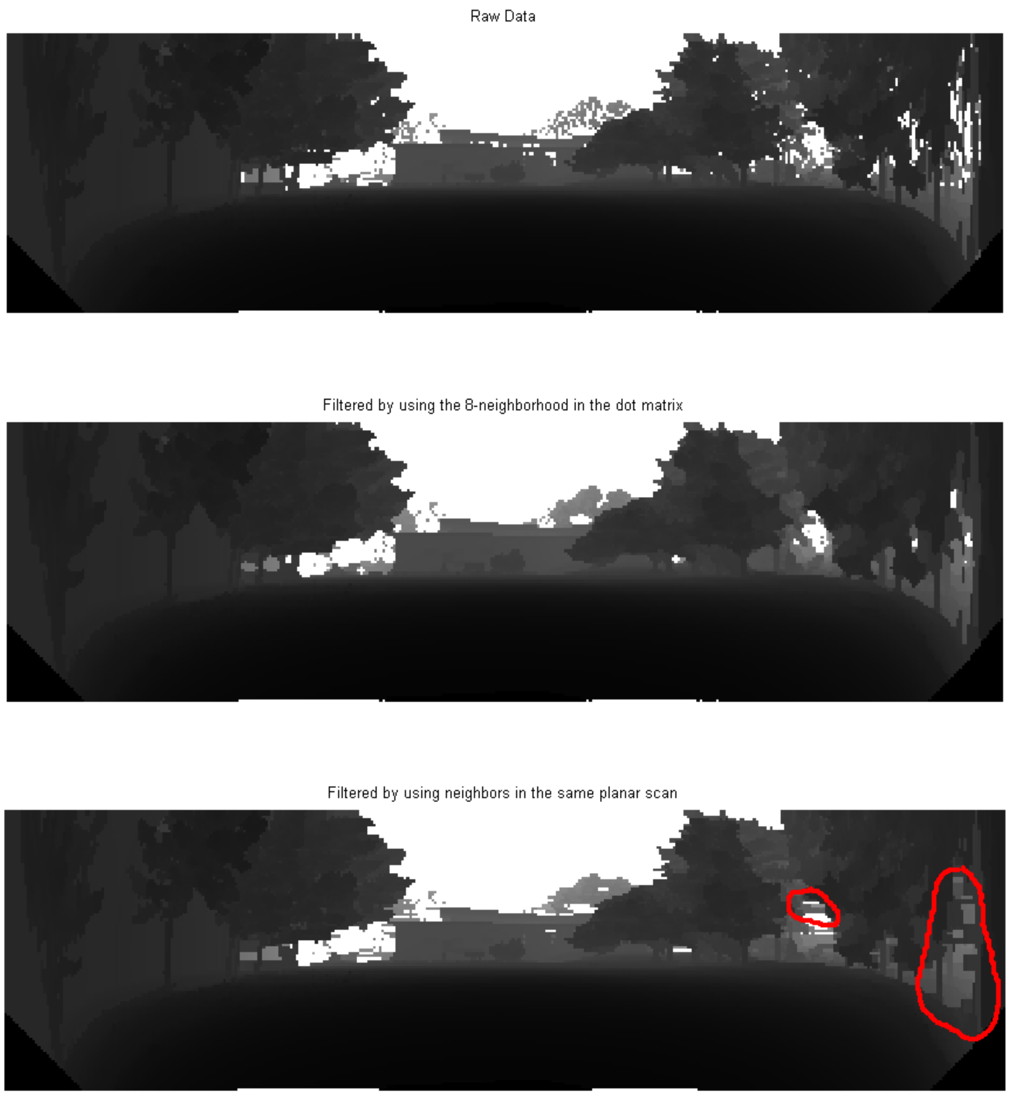

Raw Data
Filtered by using the 8-neighborhood in the dot matrix
Filtered by using neighbors in the same planar scan

Figure 4.7 Top: Raw data, Middle: The current filter is applied, Bottom: Filter in [3] is applied

4.2.2.2 Calculation of Normals

During registration step, for each sampled point in the data scan, the corresponding points in the model scan are searched. The two correspondence criteria are the distance between possible point mates and the angle between the normals of the surfaces that the candidate points belong. These normals are calculated exploiting the fact that the data is in the form of a 101 x 361 dot matrix. Each point in Figure 4.8 represents the location of a data point in the dot matrix. The locations of points in the 4-neighborhood of the point in question are used to set the normal of the surface at that point (Figure 4.8). The normal is approximated as a unit vector in the direction of cross products of vectors formed by using the neighboring points in the vertical and horizontal directions.

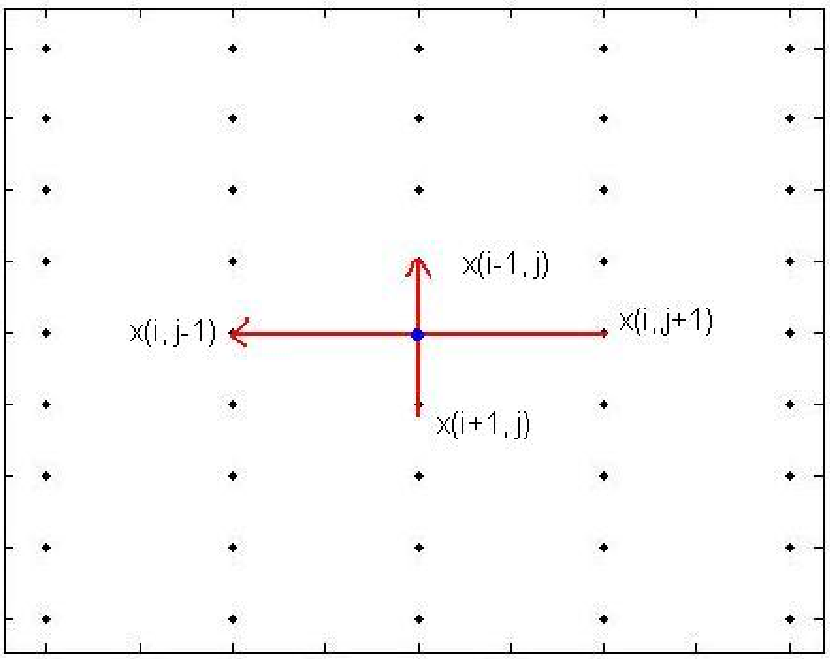


Figure 4. 8 Calculation of the normals

Let $x_{i,j}$ be the point of interest and $n_{i,j}$ be the normal of the surface at $x_{i,j}$. The index i is an integer between 1 and 101 indicating to which planar scan the point belongs to, and the index *j* is an integer between 1 and 361 indicating the place of the point in *i*'th planar scan. Then, the unit vector $n_{i,j}$ is calculated as follows:

$$n_{i,j} = ((x_{i-1,j} - x_{i+1,j}) \times (x_{i,j-1} - x_{i,j+1})) / \left\| (x_{i-1,j} - x_{i+1,j}) \times (x_{i,j-1} - x_{i,j+1}) \right\| \quad \textbf{(4.2.1)}$$

Despite the fact that this vector does not reflect the true normal of the surface, it is a good approximation to be used in the search for the correspondence. Figure 4.9 presents an example of the normals calculated in a sample 3D scan. It is observed that in surfaces of man-made structures like buildings, normals point in the same direction, which indicates that calculation conducted is a good approximation of the normals. During the matching trials, it is found out that using these normals as matching criteria always gave better results than a registration process solely depending on closest point matching.

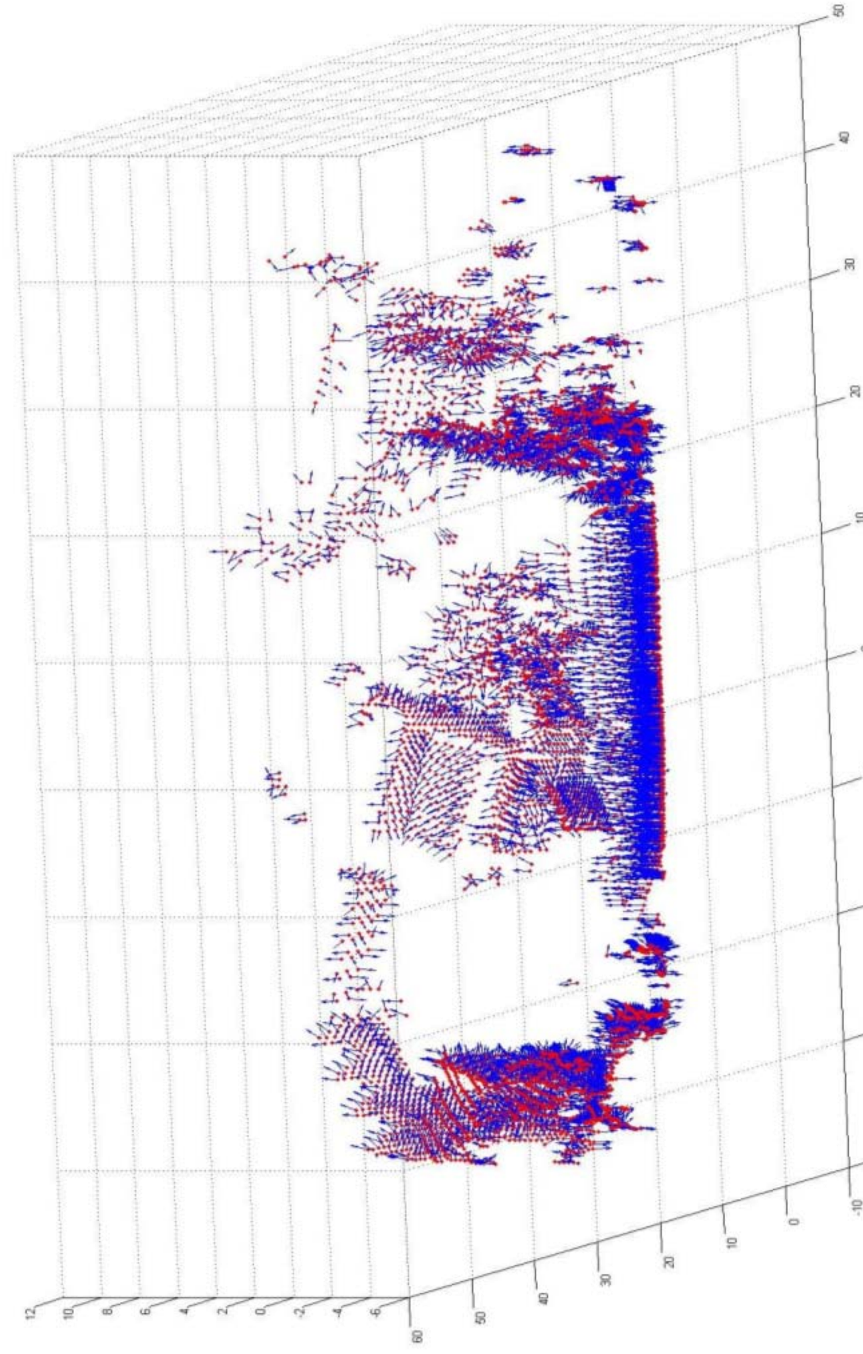


Figure 4.9 Normal directions calculated at each point

4.2.2.3 Point Reduction

Based on the idea in [3], a point reduction procedure is followed after filtering out the pepper and salt noise. According to [3], reducing points which are close to each other to one point decreases the Gaussian noise. The algorithm works as follows: Starting from the first point in a scan slice, the distance between the data point and its neighbor is calculated. If they are closer than a certain threshold, the two data points are replaced by their midpoint. The threshold is selected as 0.5m, empirically. Therefore, there is at least 0.5m distance between each point in a scan slice (a planar scan).

This procedure is applied to all the points in a scan. This reducing procedure reduces the number of points in a scan, without the loss of features (Figure 4.10). Since there are less points in the final point cloud, this reducing procedure also reduces the computation time needed for scan matching.

After the points are reduced according to the distance to their neighbors, finally, the points having the distance 81.89m to the origin are deleted from the scan cloud, since, as mentioned before, this distance value means that there are no corresponding objects in the scanning range (Figure 4.10). Not deleting these points causes false pairwise registration of two consecutive scans.

A scan slice consists of 361 distance values before and after the median filter is applied. In Figure 4.10, it is observed that although the reduced scan has 124 data points, the shape of the scan is preserved. Using this ability to present the same shape with less points, a great deal of computational time is gained during point matching. The overall effect of the point-reducer on the 3D point cloud can be observed in Figures 4.11 and 4.12.

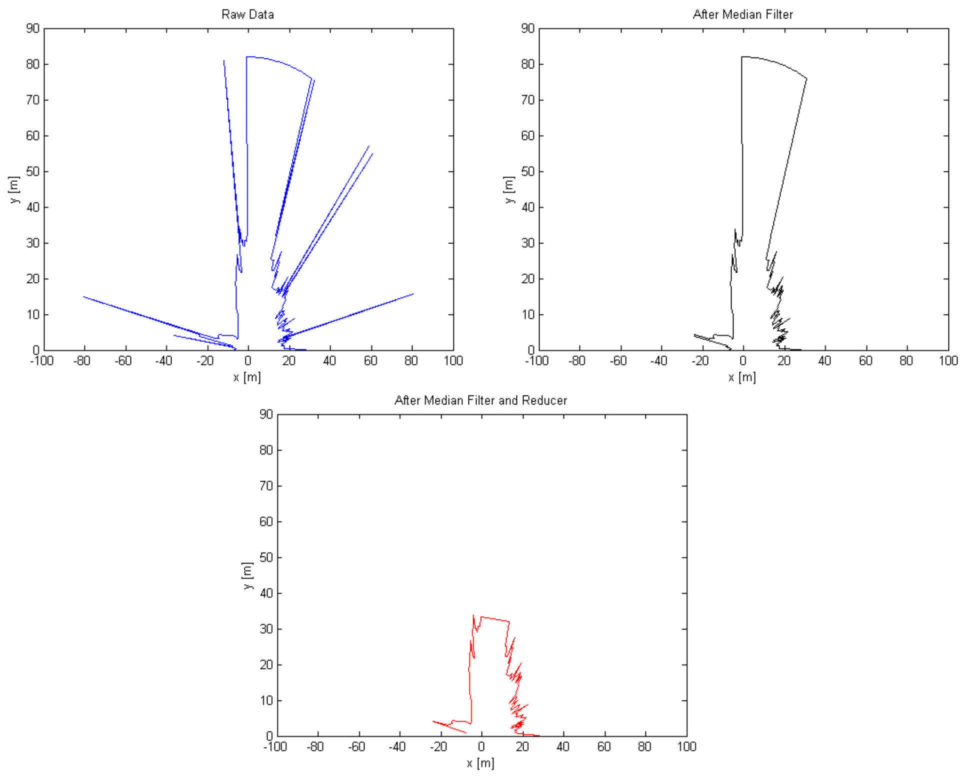


Figure 4.10 Despite reducing the number of points, the general structure of the scan is preserved.

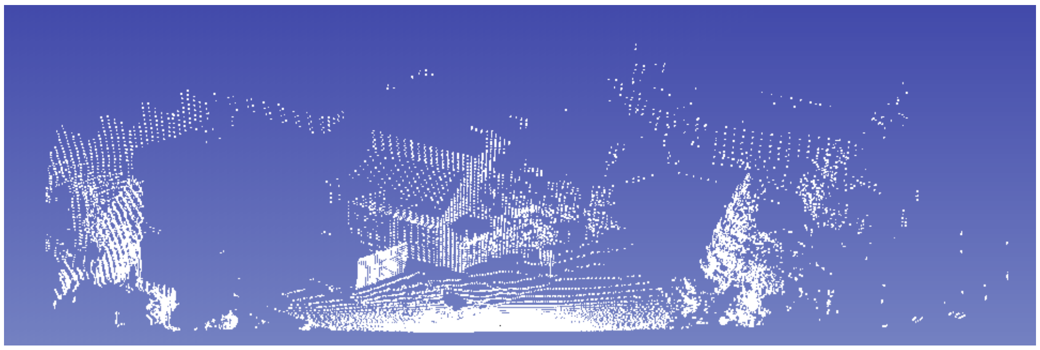


Figure 4.11 3D point cloud collected from car park of Mechanical Engineering, G building. Median filter is applied, but points are not reduced and the point cloud has 36461 data points.

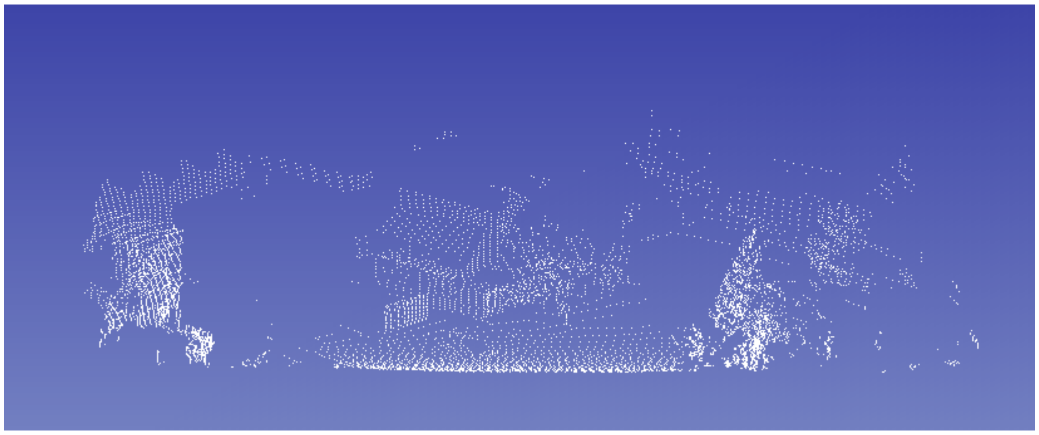


Figure 4.12 The same point data in Figure 4.11 is reduced. With much fewer points (8439), the same structures are represented.

4.2.3 Search for Nearest Neighbors

The most time-consuming, computationally expensive step of ICP algorithm is finding the corresponding points between the points of the data set and the model set [35], [36], [37], [41]. Although the point number is reduced in the preprocessing step, there is still a need for a fast search algorithm, instead of brute-force searching, which is computationally very costly. In brute-force searching, a corresponding point for each sampled point in the data set is found by comparing their distance to every point in the model set. This results in a computational complexity in the order $O(N_D N_M)$, where N_D is the number of sampled points from the data set and N_M is the number of points in the model set.

In this study, the well-known kD-tree structure, which was first introduced in [42], is used for closest point searching. The kD-tree is a data structure formed by splitting the search space by planes orthogonal to each of the coordinate axes. The splitting planes are originated from nodes of the structure, which are, in fact, data points in the search space. The splitting is continued until each bucket contains 10 or less points. The bucket size of 10 is the most efficient size for the kD-Tree search, as [3] suggests. An example of a kD-tree formed in 3D space is presented in Figure 4.13.

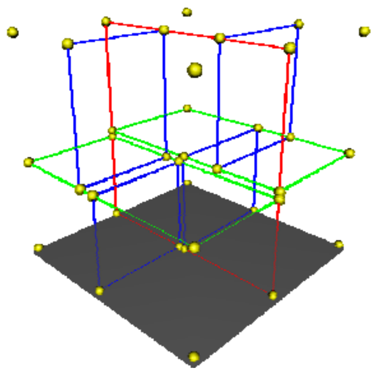


Figure 4.13 An example of an 3 dimensional kD-tree. First, the cube is split by the red plane at the root node. Second, corresponding left-child and the right-child are split into two volumes by the green planes. Third, all the corresponding children are split by blue planes. This process goes on until no splitting node is available(that is the bucket size is 1). [43]

Before starting the ICP iteration, a kD-tree structure for the model set is formed. The pseudo code for forming the kD-tree is given in Figure 4.14. The idea behind splitting the search space into regions is simple: if the splitting coordinate value of the point at hand is greater than the splitting coordinate value of the node at the splitting axis, there is no need to search the points in the other side of the splitting axis (or vice versa). The search algorithm is adapted from [52] and the pseudo-code is given in Figure 4.15. For the details of this algorithm, one may refer to [52].

```
Function  node kDTree (pointList, depth)

if pointList.Size<maxBucketSize
        node is leaf
        return
else
        axis=depth mod space-dimension
        sort pointList according to (axis)-coordinate values
        selectedpoint=select midpoint from sorted list
        node.axis=axis  //the splitting axis
        node.location=selectedpoint(x,y,z)
        node.leftchild=kDTree(points in pointList |  point(axis)<node.location(axis))
        node.rightchild=kDTree(points in pointList |  point(axis)=>node.location(axis))
end
```

Figure 4.14 Pseudo-code for constructing the kD-tree

```
global  nndist=infinity
global neighbor=searchtree.rootnode.location

Function  (neighbor,nndist)= searchNN(searchtree, point, nndist, neighbor)

if (searchtree is not leaf)
        value=searchtree.location(searchtree.axis)
        pointx=point(searchtree.axis)
        if pointx<value
                (neighbor,nndist)=searchNN(searchtree.leftchild ,point ,nndist ,neighbor)
                if (pointx+nndist>value)
                 (neighbor,nndist)=searchNN(searchtree.rightchild ,point ,nndist ,neighbor)
                end
        else
                (neighbor,nndist)=searchNN(searchtree.rightchild,point,nndist,neighbor)
                if (pointx-nndist<val)
                 (neighbor,nndist)=searchNN(searchtree.leftchild,point,nndist,neighbor)
                end
        end
else
        foreach (point x in node)
                dist=distance(x , point)
                if (dist<nndist)
                        nndist=dist
                        neighbor=x
                        val=searchtree.location(searchtree.axis)
                end
        end
end
```

Figure 4.15 Pseudo-code for nearest-neighbor search algorithm.

4.2.4 Point Matching

In order to compute the transformation between two successive 3D point-clouds returned by the sensory system, one must find the matching points between two point clouds. This matching procedure is the heart of the ICP algorithm, and there are many different techniques in the literature, each using different constraints, for finding matching pairs. Among these are the point-to-point distance [35], [36], normal-shooting [37], comparing angles between tangents or normals [36], comparing colors [44] or intensity values [45] of the data points.

In this study point-to-point distance and the angle between normals of surfaces at matching points are used as constraints. Calculation of these normals is explained in Section 4.2.2.2.

During point matching, a corresponding point in the model set is found for each "sampled point" in the data set. These corresponding points are found by using the kD-tree of the model set. First, from the kD-tree structure of the model set, the bucket which includes the closest point to the query point in the data set is found. Then, for each point in this bucket, the distance to the query point, and the angles between the normals at these points are calculated. Among these points in the model set, the candidate with the best normal is selected as the matching point to the query point. By "best normal" it is understood that the angle between normals of matching points is less than those of all the other candidates. It can be easily shown that the angle between normals at the matching points must not exceed the angle between axes of two successive frames. So, let x_i^M be the i'th point in the model set and x_j^D be the j'th point in the data set. Then, these points are "corresponding points" only if the following conditions are satisfied:

- x_i^M lies in the same bucket of the kD-tee with the closest point to x_j^D

- $\left\| x_j^D - x_i^M \right\| < \Delta_{\max}$, where $\Delta_{\max}$ is the predefined maximum allowed distance between corresponding points. For selection of the parameter $\Delta_{\max}$, one may refer to [36].
- The angle between n_i^M and n_j^D is less than the angle between the axes of the two frames and it is less than those of all the other candidates in the bucket.

After a set of matching point pairs are formed using the above constraints, this set is updated by comparing the average distance between matching pairs with a preset threshold. For details of this analysis, one may refer to [36].

While searching for correspondences, not all the points in the data set are used. In order to speed up the iteration process, a sample set of points are selected from the data set. The methods of selection of the points are discussed in [41]. Among these methods are using all available points [35], uniform sampling [46], random sampling [47], sampling according to distribution of normals [41]. In the current study, uniform sampling strategy is used.

4.2.5 Computing the Motion

Using the updated matching pair set, one may find the motion bringing two successive point sets closer. There are four known efficient methods to compute this motion, each of which is the solution of the optimization problem trying to find the rigid transformation that minimizes the cost function in **(4.1.1)**. These four methods are using *Singular Value Decomposition (SVD)* [48] , using *Orthonormal Matrices(OM)* [49], using *Unit Quaternions (UQ)* [50], using *Dual Quaternions (DQ)* [51] .

A good quantitative comparison of these four methods is presented by Lorusso et. al in [38]. It is stated in Lorusso’s work that in terms of accuracy, there is not a significant difference between using each of these methods. In terms of stability, SVD and UQ methods are stated to be more stable. It is concluded in Lorusso’s work that SVD algorithm provides the best overall accuracy and stability, whereas it may not be as efficient as DQ for large data sets.

Although the ICP algorithm used in this work is mostly based on Zhang's [36] approach, which uses DQ while computing the motion, SVD approach will be used in this thesis. The following parts of this section explains the SVD method, proposed by Arun, Huang and Blostein [48].

Let (p_i^D, p_i^M) be corresponding point pairs, where $i = (1, 2, ..., n)$ with n being the number of corresponding point pairs, p_i^D is a point described in a coordinate frame before motion, and p_i^M is the corresponding point to p_i^D, described in a coordinate frame after motion.

The centroids of the correspondent points in their own frame are:

$$\overline{p}^D = \frac{1}{n} \cdot \sum_{i=1}^{n} p_i^D \tag{4.2.2}$$

$$\overline{p}^M = \frac{1}{n} \cdot \sum_{i=1}^{n} p_i^M \tag{4.2.3}$$

Centering corresponding points in a common coordinate frame:

$$\tilde{p}_i^D = p_i^D - \overline{p}^D \tag{4.2.4}$$

$$\tilde{p}_i^M = p_i^M - \overline{p}^M \tag{4.2.5}$$

The singular value decomposition of the cross-correlation matrix of the centered data is defined as:

$$\sum_{i=1}^{n} \tilde{p}_i^M \cdot (\tilde{p}_i^D)^T = U \cdot W \cdot V^T \tag{4.2.6}$$

Then, the rotation matrix representing the motion between the point pairs is estimated as :

$$R = V \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \det(V \cdot U^T) \end{bmatrix} \cdot U^T \qquad \textbf{(4.2.7)}$$

Finally, by using this rotation matrix, the translation between the scans can be estimated as:

$$t = \overline{p}^D - R \cdot \overline{p}^M \qquad \textbf{(4.2.8)}$$

This transformation (R, t) is applied to all the points in the data set. This brings the two point clouds "closer". Then, if the error defined in **(4.1.1)** is still above a predefined value the iteration continues until the terminating condition occurs.

The iteration process for finding the motion between two robot poses is presented in this chapter. For an overview of the algorithm, see the flowchart presented in Figure 4.18. A result of a registration process is also shown in Figure 4.16 and Figure 4.17. Different colors in these figures represent data acquired from two different poses of the experimental setup. The graphical engine developed by Yarkınoğlu [54] is used to visualize the 3D data.

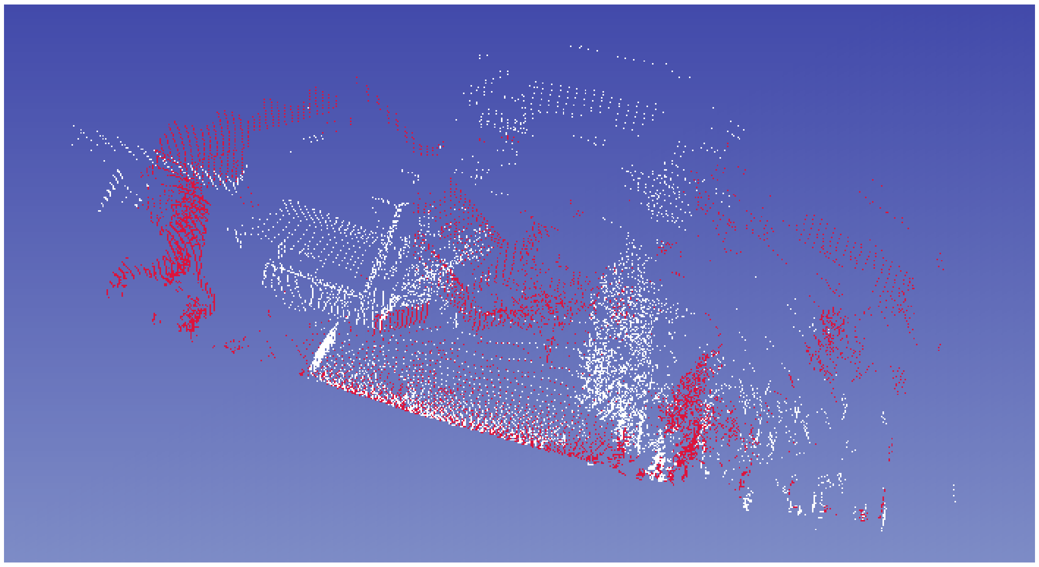


Figure 4.16 Two 3D scans, represented in the same coordinate frame before registration

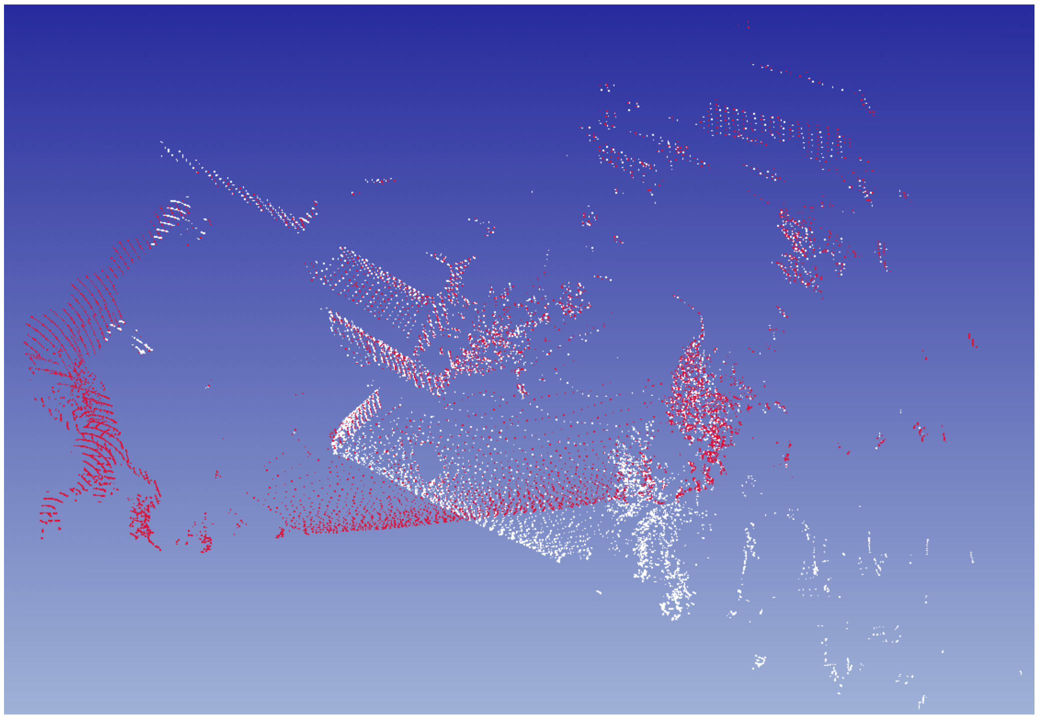


Figure 4.17 The 3D scans in Figure 4.13 are registered by using the ICP algorithm.

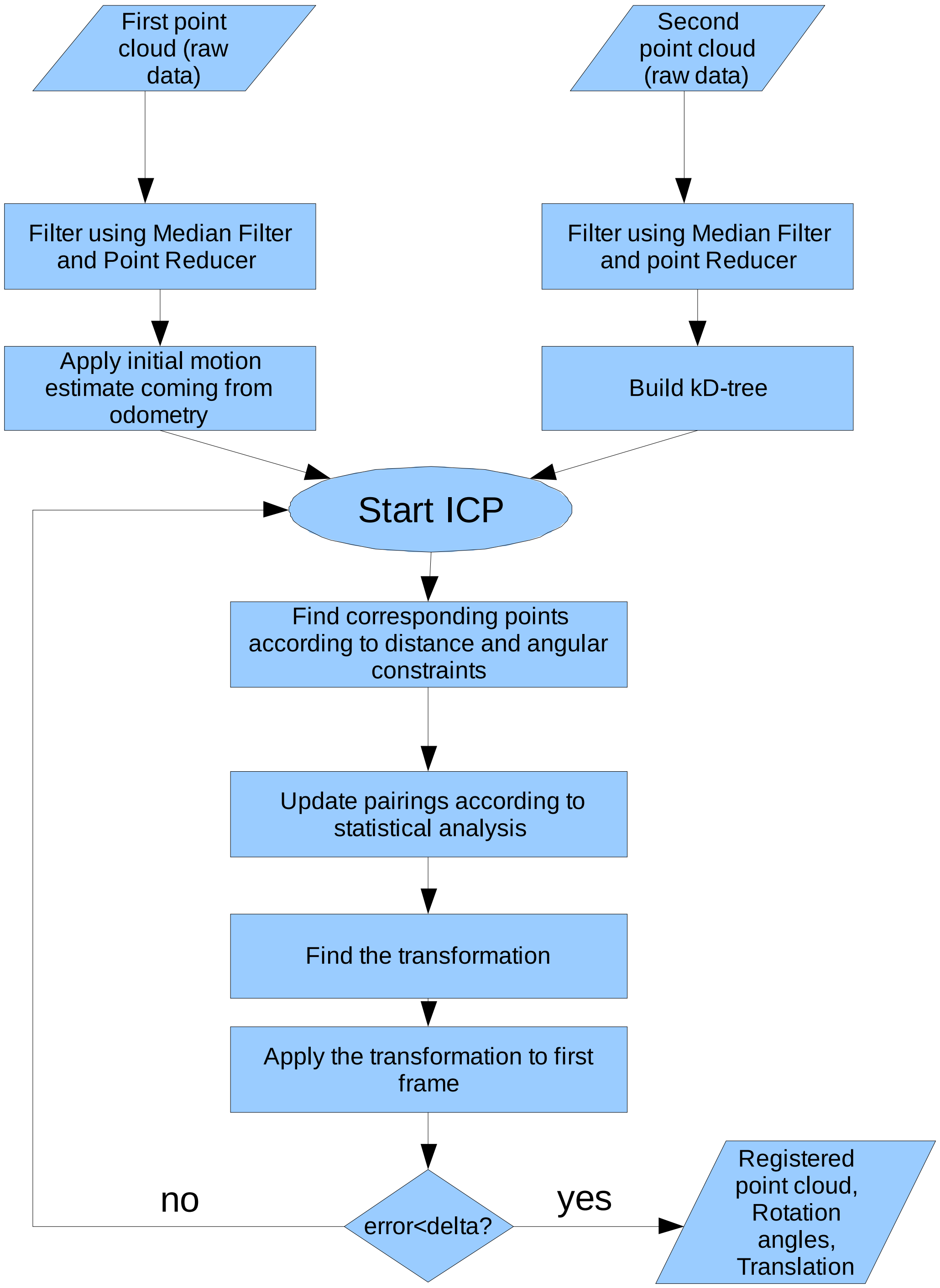

First point cloud (raw data)
Second point cloud (raw data)
Filter using Median Filter and Point Reducer
Filter using Median Filter and point Reducer
Apply initial motion estimate coming from odometry
Build kD-tree
Start ICP
Find corresponding points according to distance and angular constraints
Update pairings according to statistical analysis
Find the transformation
Apply the transformation to first frame
error<delta?
no
yes
Registered point cloud, Rotation angles, Translation

Figure 4.18 Flowchart for the registration process

CHAPTER 5

ROBOT LOCALIZATION

The 3D scans acquired at different positions of the robot are used to form the map of the environment. This is done by merging the scans into a global coordinate system. If the robot were precisely localized, this registration could be done by directly applying robot motion to the scans. However, due to the imprecise robot sensors, self localization is erroneous, so the geometric structure of overlapping 3D scans has to be considered for registration [23]. The details of this registration process were given in the previous chapter. This scan registration process not only results in forming a map of the environment, but the results of the registration process is also used for localizing the robot relative to the starting position.

5.1 Calculation of the Robot Pose

After the registration of two 3D point clouds, a transformation matrix $H_{i,i+1}$, representing the Affine Transformation transforming the points in the (i+1)th scan into the coordinates of i'th scan, is obtained. $H_{i,i+1}$ is given in equation (5.1.1), where $R_{i,i+1}$ is the 3 x 3 rotation matrix representing the rotation between i'th and (i+1)th coordinate frames, and $t_{i,i+1}$ is the 3 x 1 translation matrix representing the translation between these two frames.

$$H_{i,i+1} = \begin{bmatrix} & R_{i,i+1} & & t_{i,i+1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \textbf{(5.1.1)}$$

For each scan pair, there is one such transformation matrix obtained. The poses of the robot can be obtained by using these transformation matrices. Let starting position of the robot be $r_0 = (0,0,0)$ and the unit normal vector representing the heading of the robot be $h_0 = (0,1,0)$. The relations between consecutive frames are represented in Figure 5.1. The robot poses can be calculated as:

$$r_1 = r_1^{\,0} = H_{0,1} \cdot o_1 \qquad \textbf{(5.1.2)}$$

$$r_2^{\,1} = H_{1,2} \cdot o_2 \qquad \textbf{(5.1.3)}$$

Then;

$$r_2 = r_2^{\,0} = H_{0,1} \cdot r_2^{\,1} = H_{0,1} \cdot H_{1,2} \cdot o_2 \qquad \textbf{(5.1.4)}$$

Where o_i represent the origin of the i'th frame. That is, $o_i = (0,0,0)$. Then, i'th position of the robot is calculated as;

$$r_i = H_{0,1} \cdot H_{1,2} \cdot H_{2,3} \cdot \ldots\ldots\ldots\ldots \cdot H_{i-1,i} \cdot o_i \qquad \textbf{(5.1.5)}$$

The heading of the robot at each pose can be calculated as;

$$h_1^{\,0} = R_{0,1} \cdot h_1^{\,1} \qquad \textbf{(5.1.6)}$$

$$h_2^{\,1} = R_{1,2} \cdot h_2^{\,2} \qquad \textbf{(5.1.7)}$$

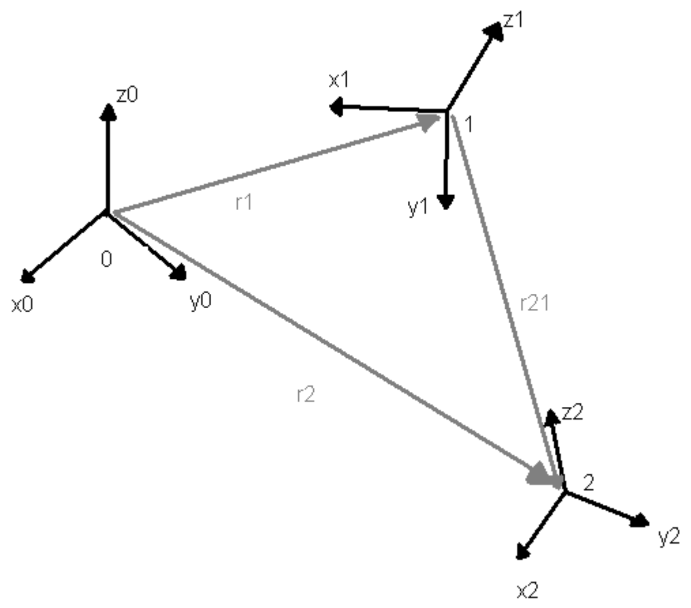


Figure 5.1 Finding the relative motion between consecutive robot poses.

$$h_2^{\ 0} = R_{0,1} \cdot h_2^{\ 1} = R_{0,1} \cdot R_{1,2} \cdot h_2^{\ 2} \qquad \textbf{(5.1.8)}$$

$$h_i^{\ 0} = R_{0,1} \cdot R_{1,2} \cdot ....R_{i-1,i} \cdot h_i^{\ i} \qquad \textbf{(5.1.9)}$$

Where $h_i^{\ j}$ is the heading of the robot at i'th position, represented in the j'th coordinate frame, and $h_0^{\ 0} = h_1^{\ 1} = h_2^{\ 2} = ... = h_i^{\ i} = (0,1,0)$.

5.2 Results

While the experiments for testing the localization procedure were conducted, a real robot was not present, so the experiments were made by moving the range finder on a table to various positions and collecting range data. Because of the lack of the robot, instead of using encoders, a tape measure was used to measure position changes between different poses of the scanner.

During the localization process, the starting position is always taken as $r_0 = (0,0,0)$ and the heading vector at starting position is always taken as $h_0 = (0,1,0)$. The measurements taken while conducting the first experiment is given in Table 5.1. The values under heading columns represent the x, y and z components of the unit vector in the heading direction of the robot.

Table 5.1 Measurements obtained in the first experiment

	Position X	Position Y	Position Z	Heading X	Heading Y	Heading Z
0	0	0	0	0	1	0
1	0	0	0	-0.707	0.707	0
2	0	4	0	0	1	0
3	0	6	0	0	1	0
4	0	6	0	0.707	0.707	0

The poses presented in Table 5.1 were obtained by using a tape measure, and these values are used as the odometry input to the registration process. Figure 5.2 presents the path followed while taking the data for this first experiment.

Obviously, since poor measuring was performed during the collection of the data, the path and heading vectors presented in Figure 5.2 do not represent the true values. The values obtained from the outputs of the pairwise registrations for these 5 poses are given in Table 5.2 and Figure 5.3.

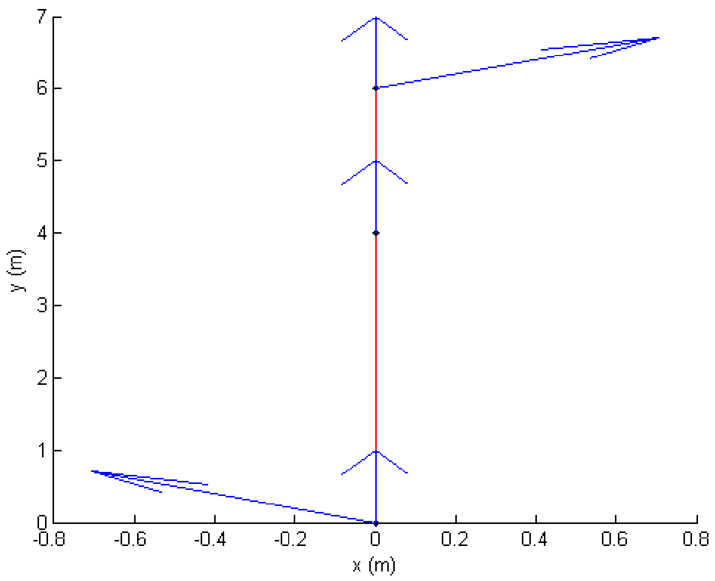


Figure 5.2 The measured path in the first experiment. The red lines denote the path followed, black dots denote the poses where 3D data is acquired and the arrows are the heading angles.

Table 5.2. Poses and headings calculated by using point cloud registration for the experiment given in Table 5.1

Data #	Position X	Position Y	Position Z	Heading X	Heading Y	Heading Z
0	0	0	0	0	1	0
1	-0.15	-0.07	0	0.603065	0.796474	0.044066
2	0.34	3.67	-0.14	0.018828	0.999814	0.004127
3	0.48	6.05	-0.16	0.011945	0.999929	0.000166
4	0.96	5.94	-0.22	0.668880	0.743350	0.005261

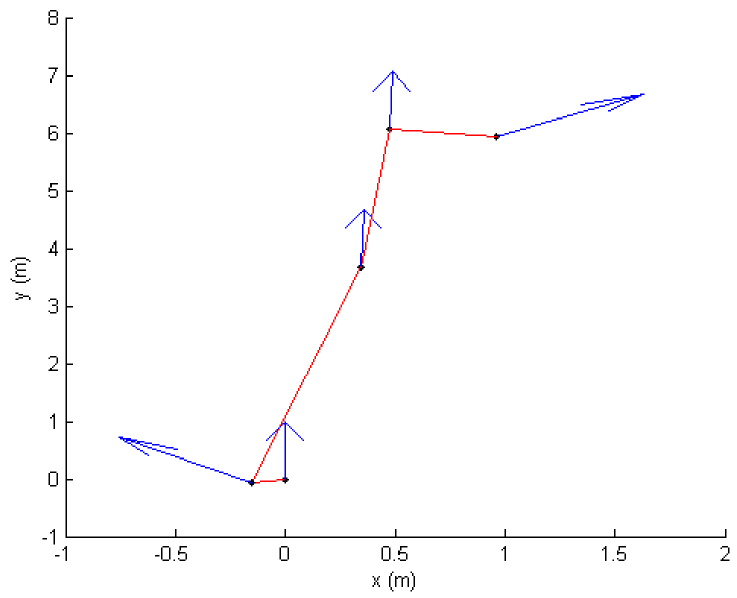


Figure 5.3 Localization obtained after using the outputs of the pairwise matchings.

One may compare the reliability of these two pose data by comparing the maps formed by using robot poses in each localization data. Figure 5.4 presents the top view of the map formed if 3D scans were registered on one each other by only using the odometry data. In Figure 5.4, different colors represent range data acquired at different poses. One can easily observe that the map is inconsistent and not usable. The satellite image of the experiment area is given in Figure 5.5.

For comparison purposes, the top view of the map formed by using range image registration is given in Figure 5.6. As observed in the figure, the map is consistent and represents the characteristics of the environment.

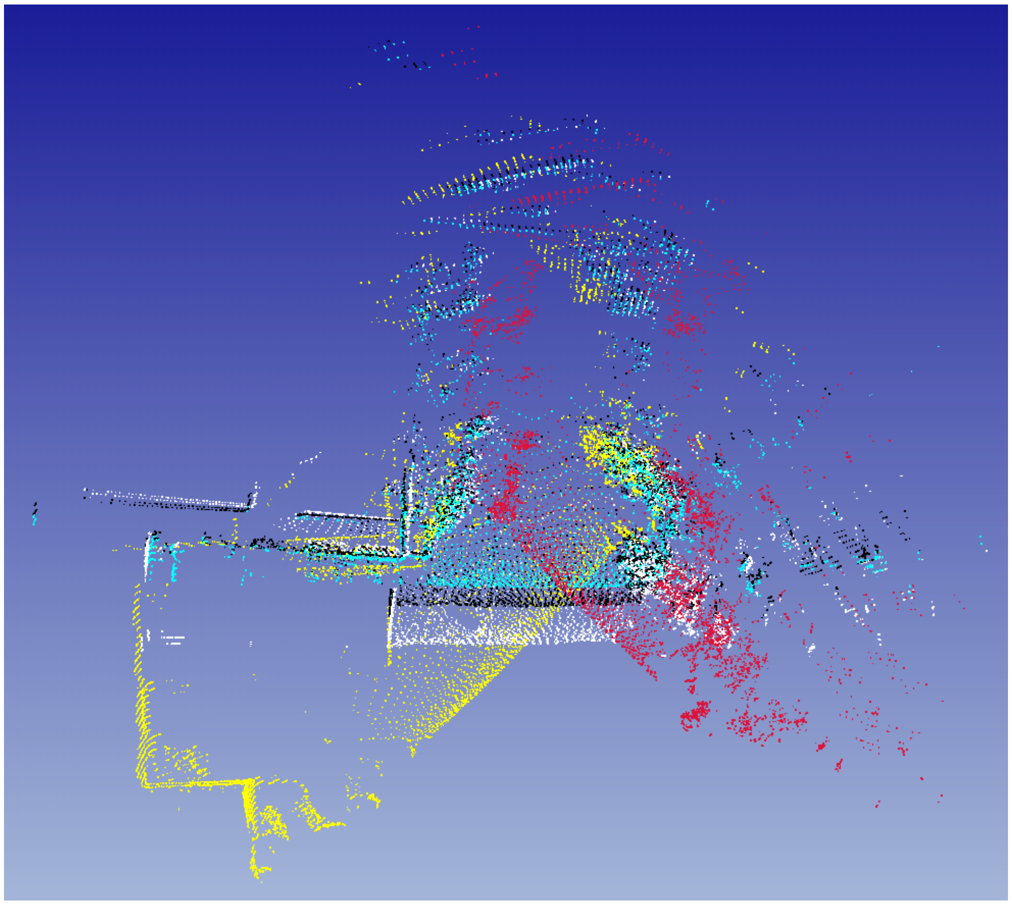


Figure 5.4 The map formed by using solely odometry data.

It must be noted that even after the pairwise registration processes are finished and robot poses are found relative to the starting position, there will be small errors associated with the resulting map, and hence the robot localization. This errors rise from the fact that each pairwise registration's precision is limited, and the error accumulates when all the scans are registered one after another. The solution to this problem is using multi-view registration and will be discussed in Chapter 6.

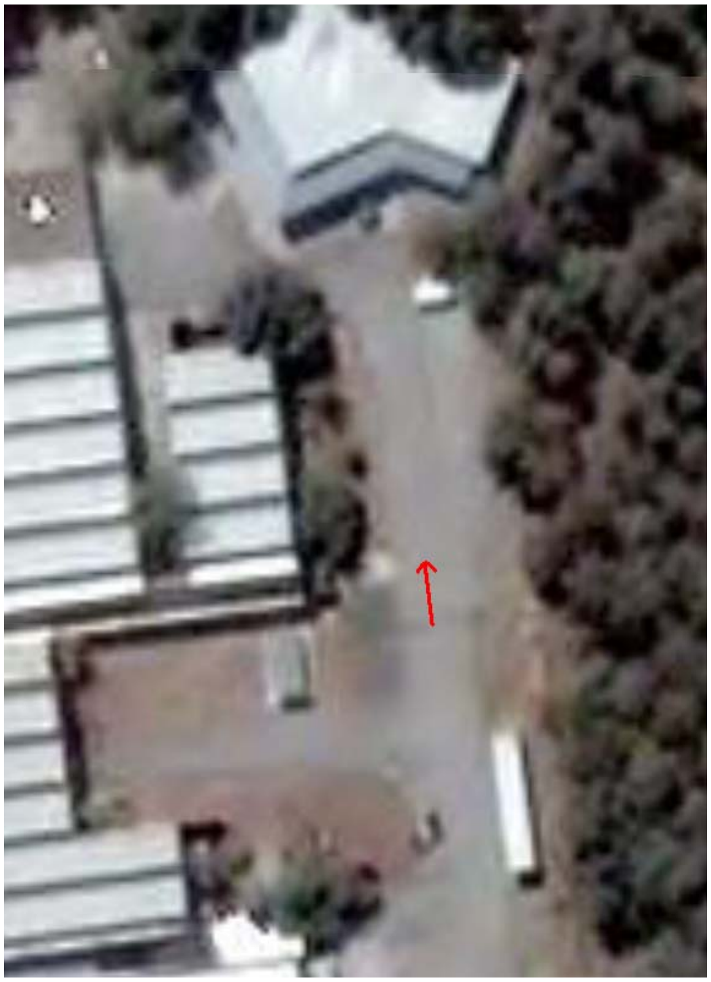


Figure 5.5 The satellite image of the area that the range data is collected. Red arrow denotes the starting position for Experiment 1.

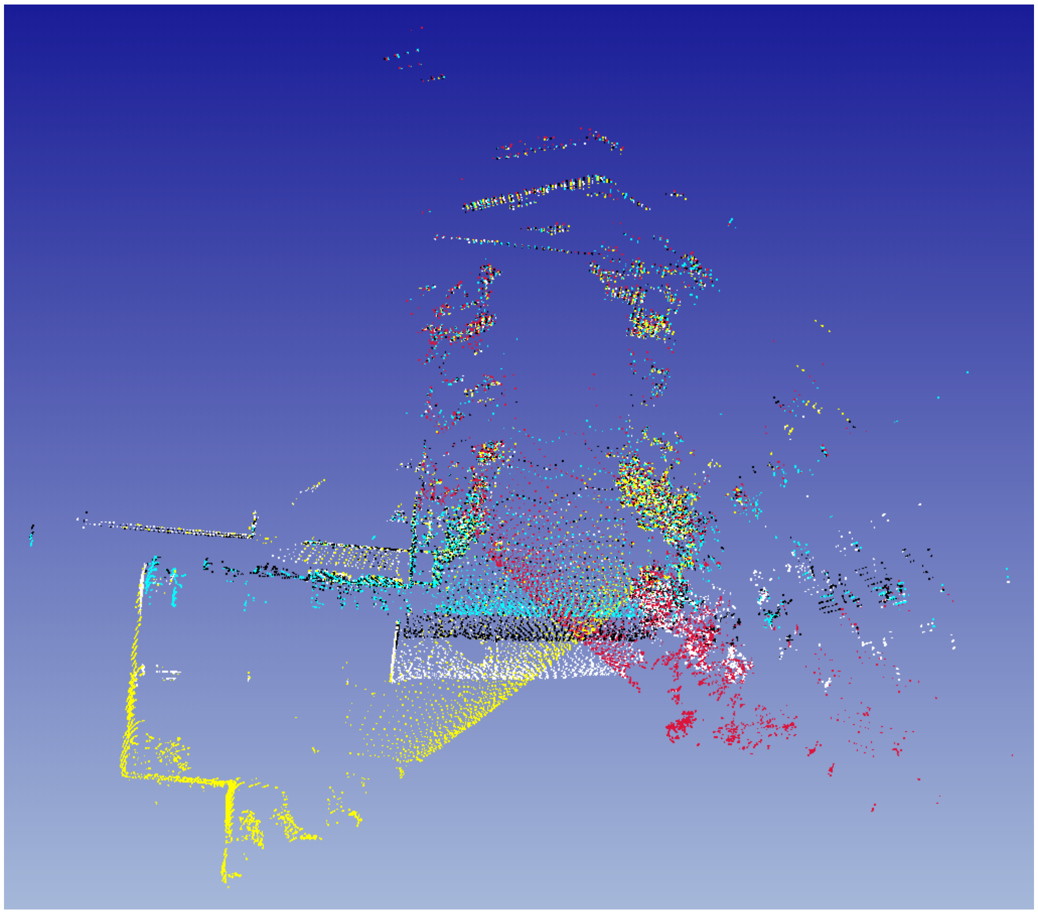


Figure 5. 6 Map formed by pairwise registration.

CHAPTER 6

MULTIVIEW REGISTRATION

After the pair-wise registrations are completed, all scans must be represented in a global coordinate frame in order to form the consistent digital map of the environment that the robot navigates through.

6.1 Multiview Registration Algorithms

One method to form the global map is directly using the outputs of the pair-wise registrations and directly connecting them one after another, since the transformations between consecutive scans are known. However, as stated in the previous chapter, there will still be some errors in the map formed and in the robot localization. These errors are due to the fact that not all pair-wise registrations are perfect. These errors accumulate and may lead to inconsistent maps when number of the scans increase. A similar approach to this technique was proposed by Chen and Mendioni [37], for object modeling, where after each pair-wise registration step, registered data is merged into one “metascan”, which is a point cloud composed of all the previously registered scans and new data is registered on this metascan each time. These two methods are called “Sequential Registration” in the literature and they both suffer from error accumulation [3],[55].

The alternative to sequential registration is “Simultaneous Registration” (also known as “Global Registration” or “Multiview Registration”), which uses the fact that new data may have overlaps with the previously registered data end these overlaps may be used to minimize the errors arouse during pair-wise registration. In [55], Nishino and Ikeuchi use an extension of the pair-wise ICP algorithm to handle multiple range images simultaneously. A data set is defined as the particular range image in interest and scene is defined as the one of the remaining range images in the range image set. Point correspondence search is performed between the data set and the scene. After finding point mates for all of the range images and calculating the corresponding transformations, all the transformations are applied simultaneously to corresponding range images. A similar approach was developed by Pulli [56]. The important difference of Pulli's algorithm is, range images are added one at a time to a set of views, in order not to get stuck in a local minimum due to unfavorable initial configuration. That is, not all views are used for point correspondence search unlike [55]. An approach very similar to [56] was used for global map forming and localization in [3], [23] and [25]. In [57] genetic algorithms were used for object modeling by using multi view registration. A method of obtaining connectivity graphs between views was proposed by Matabosch et. al. [58] to use image registration for building object models. By using the connectivity graph, and assigning the registration errors obtained between range images as the costs of the graph, optimal registration is searched by using Dijkstra's shortest path algorithm. Lu and Milios [59] form a network of relations between range scans and apply a procedure based on maximum likelihood to combine these relations and localize the robot in a planar environment. In [60], a network of views is constructed and the view with most links to other views is called the central view. This central view is not allowed to transform through the global registration. Then, to diffuse registration errors, each non-central view is registered onto the central view.

6.2 Current Study

In this study, an algorithm very similar to the approach followed in [56] is used for reducing the global error associated with the global 3D map of the environment. The difference is that in the current study, the first scan is always fixed and does not change its position during the registration process. This is due to the fact that the first scan sets the coordinate frame of the global map.

The reason for choosing Pulli's method for global registration is that instead of using all neighboring views to correct a view's position, the algorithm uses an active set into which a new neighboring scan is added after each position upgrade, instead of using all the scans for at once for global registration. This avoids the algorithm to be stuck in a local minimum of the error function and give wrong registrations, unlike other simultaneous matching algorithms do.

The algorithm for global registration, adapted from [56], is presented in Figure 6.1 and is applied as follows: After the pairwise registration step, scans are added to a static set. Then, neighboring scans for each scan in the set are calculated. Two scans are neighbors if they have more than 500 corresponding points. The point correspondences at this stage are calculated by using only the distance constraint. Points in different scans are stated to be correspondents if the distance between them is equal to or less than 0.5m. After this stage, the scan with most links (i.e. with the most neighboring scans) is removed from the static set and added to the active set. Then views in static set which have the most links into the active set are removed from the static set and added to the active set one at a time. The names active and static arise from the fact that registration process only uses the scans in the active set. At each addition of a scan to the active set, a queue is initialized with this new scan, and the ICP algorithm is used to align this scan with a model point cloud consisting of its neighboring scans in the active set. After this aligning step, if the current scan changes its position more than an expectable threshold, its neighbors in the active set are added to the end of the queue.

The function `mostNeighbors` in Figure 6.1 finds the scan from the first set which has the most number of neighbors in the second set. The function `FindNeighbors` returns the neighboring scans of the current scan in the active set.

```
Add all views to staticSet

foreach scan s in staticSet
    findNeighbors(s, (staticSet-s))
end

Current= mostNeighbors(staticSet,staticSet)
activeSet.Add(Current)
staticSet.Remove(Current)

while(staticSet.Count>0)
    current= mostNeighbors (staticSet, activeSet)
    activeSet.Add(current)
    staticSet.Remove(current)
    queue.Enqueue(current)
    while(queue.count>0)
        current=queue.Dequeue()
        neigbors=FindNeighbors(current,activeSet)
        build_kDtree(neighbors)
        (anglechange,distancechange)=ICP(current,neighbors)
        if (anglechange>thresA || distancechange>thresD)
            foreach scan s in neighbors
                if (s is not already in queue)
                    queue.Enqueue(s)
            end
        end
    end
end
```

Figure 6.1 Pseudocode for global registration algorithm.

After the “active” neighbors to the current scan are found, they form a model set and the kD-Tree structure for this model set is formed before ICP iteration. The ICP iteration aligns the current scan with the model set formed by using its neighbors in the active set. The rotation and translation components of the motion of the current scan after this alignment is returned from the ICP function. If the size of one of these changes exceeds a predefined threshold, the current scan’s neighbors in the active set are added to the end of the queue if they are not already in the queue. Thresholds are defined as `thresA=0.5°` around the rotation axis and `thresD=0.5m` along the translation direction of the alignment. Notice that the kD-tree structure must be built all over again each time, since the scans are subject to transformations in each step of the inner while loop.

6.3 Results

In order to test the global registration algorithm, another experiment was conducted. In this experiment, 3D range data were acquired at 14 different poses. The satellite image of the test site is given in Figure 6.8. In order to compare the mapping and localization success after global registration, visual inspection is conducted between the maps formed after pair-wise registration and global registration. Figure 6.2 presents the map formed by using only the measurement data in Table 6.1.

Table 6.1 Measured position changes in Exp.2

Data#	Position X	Position Y	Position Z	Heading X	Heading Y	Heading Z
0	0	0	0	0	1	0
1	1.4	2.8	0	0.7071	0.7071	0
2	2.8	2.8	0	0.7071	0.7071	0
3	4.2	4.2	0	1	0	0
4	4.2	4.2	0	0.7071	0.7071	0
5	7	7	0	0.7071	0.7071	0
6	9.9	9.9	0	0.7071	0.7071	0
7	11	11	0	0.7071	0.7071	0
8	14	14	0	0.7071	0.7071	0
9	15.5	15.5	0	0	1	0
10	15.5	15.5	0	1	0	0
11	15.5	15.5	0	0.7071	0.7071	0
12	18.4	18.4	0	0.7071	0.7071	0
13	18.4	18.4	0	1	0	0

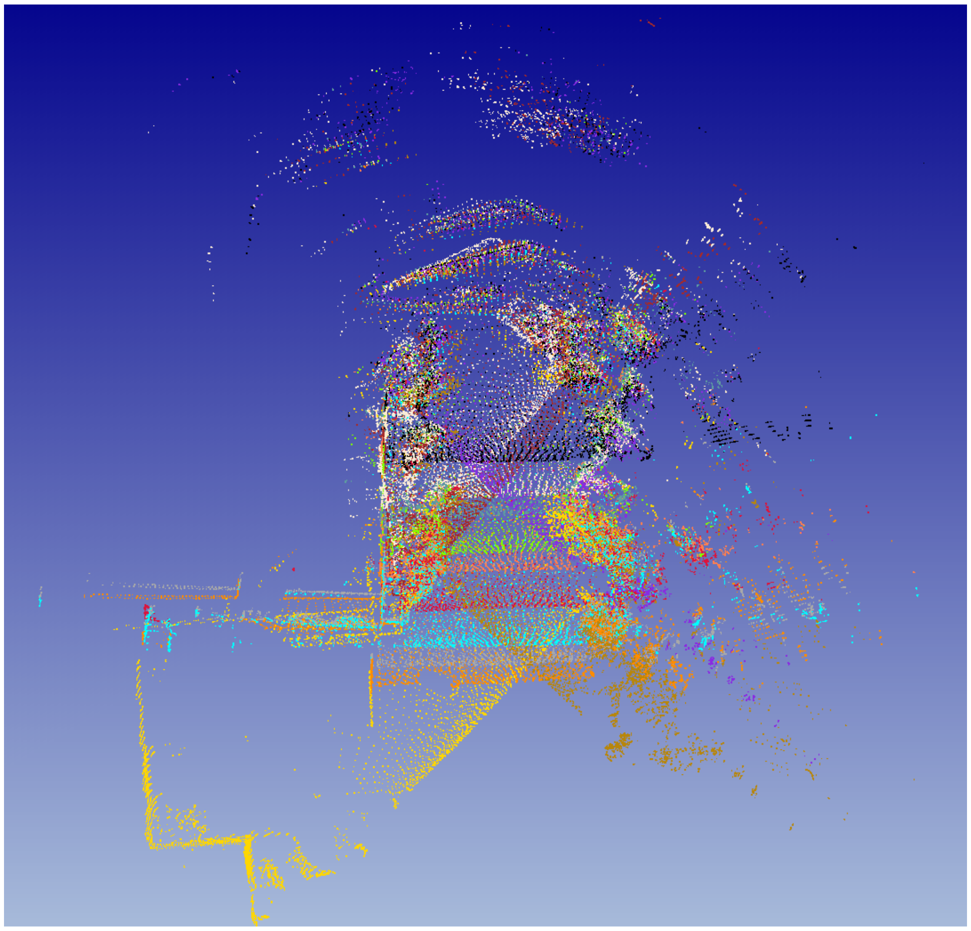


Figure 6.2 Top view of the 3D map formed by only using the odometry data for experiment 2.

As expected, using only odometry data is not sufficient for building a consistent map. The map presented in Figure 6.2 is inconsistent and do not represent the true characteristics of the environment. The estimated path by using odometry measurements is given in Figure 6.3. Figures 6.4 and 6.5 present the maps formed by using pairwise registration alone and multiview registration. Different colors in these figures represent data taken from different poses.

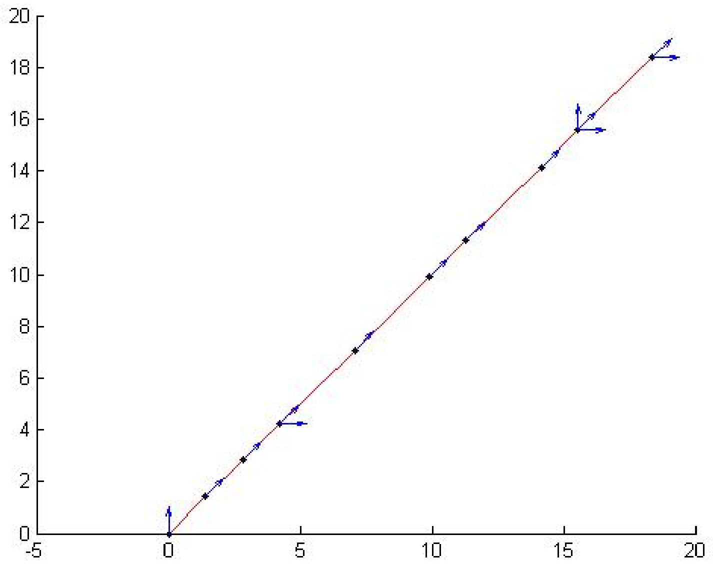


Figure 6.3 Robot poses estimated by the odometry data (Table 6.1)

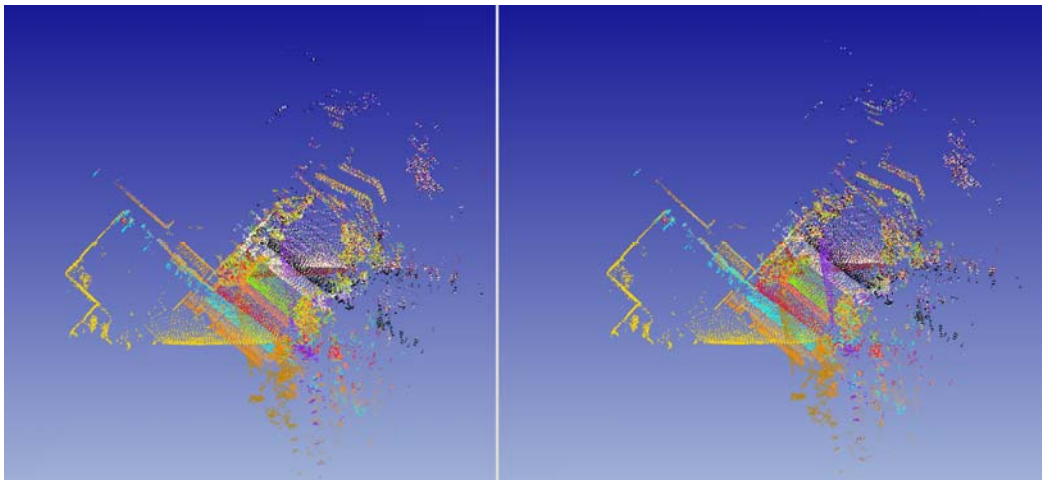


Figure 6.4 Left: Top view of the map formed after pairwise registration. Right: Top view of the map formed after global registration

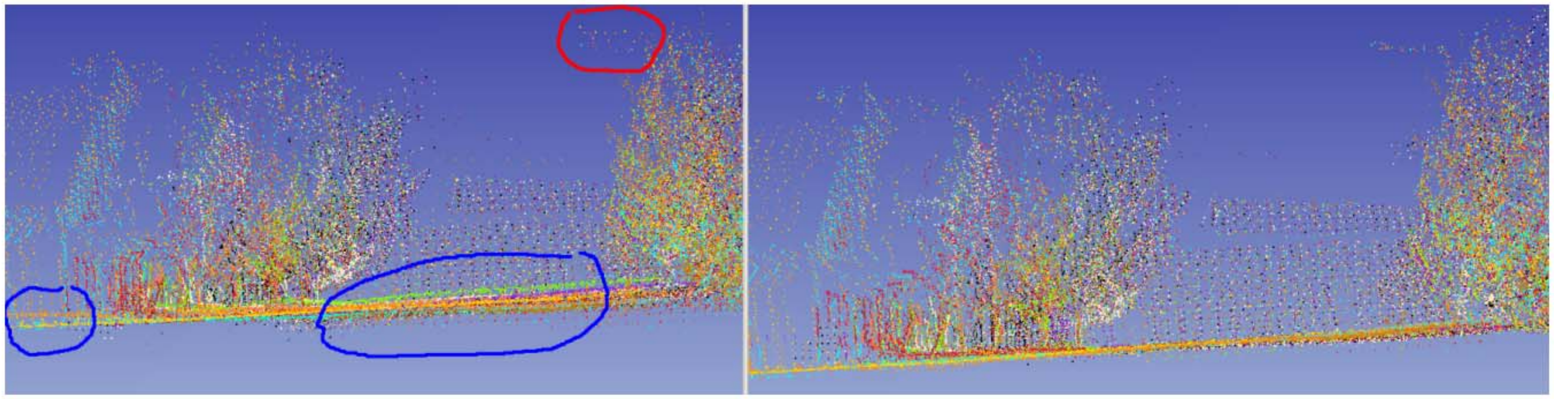


Figure 6.5 Left: Map after pairwise registrations, Right: Map after global matching

Despite there seems no significant difference between top views of maps formed after pairwise and multiview registrations, observing Figure 6.5, one may understand the improvement in the map after global registration. The red mark in the left figure encloses a street lamp. It is observed that a deformed shape occurs after pairwise registrations. Using multiview registration corrects this error (Figure 6.5, Left). More importantly, as marked by blue in Figure 6.5, there are elevation errors between data taken at different poses. A map which does not represent true characteristics of the terrain and shows discontinuous elevation jumps is formed. Figure 6.5 (Right) shows that multiview registration solved this problem.

Robot localization by using only outputs of pairwise registrations is presented in Table 6.2 and Figure 6.6. Table 6.3 along with Figure 6.7 presents the poses obtained after using multiview registration.

Table 6.2 Robot poses obtained after pairwise registrations

Data#	Position X	Position Y	Position Z	Heading X	Heading Y	Heading Z
0	0	0	0	0	1	0
1	1.35	1.25	-0.07	0.642	0.7663	-0.0259
2	2.45	2.48	-0.24	0.6298	0.7754	-0.0454
3	4.27	3.87	-0.53	0.9842	0.1769	-0.0066
4	4.04	4.24	-0.54	0.6208	0.7827	-0.0441
5	6.33	6.91	-0.78	0.5983	0.801	0.0225
6	8.97	9.53	-0.68	0.6375	0.7704	0.0018
7	10.44	11.18	-0.53	0.6335	0.7738	0.0009
8	12.78	13.72	-1.30	0.6425	0.766	0.0237
9	13.80	16.07	-1.58	-0.0033	0.9941	-0.1088
10	14.28	15.57	-1.29	0.9913	0.1256	-0.0381
11	14.18	15.46	-1.20	0.6256	0.7801	0.0104
12	16.92	18.49	-1.68	0.6135	0.7836	-0.0978
13	16.57	18.68	-1.75	0.0675	0.9931	-0.096

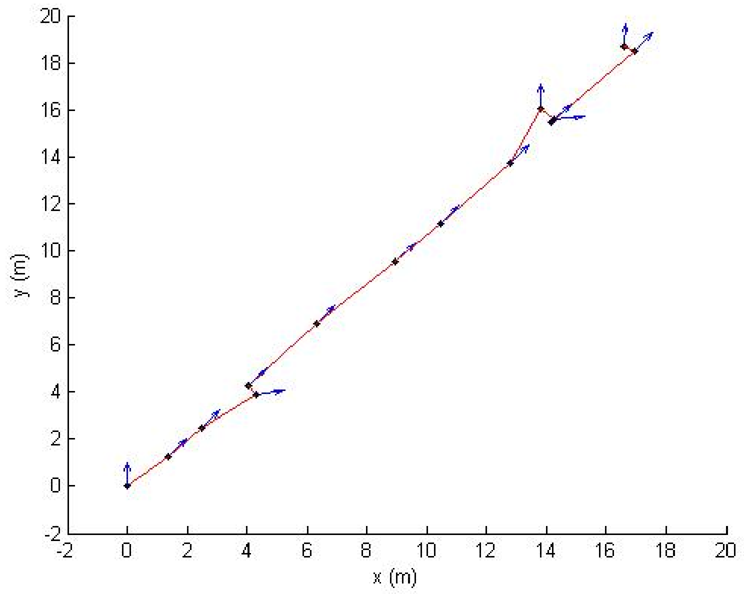


Figure 6.6 The path calculated after the pairwise registrations

.

Table 6.3 Calculated poses after multiview registration step

Data#	Position X	Position Y	Position Z	Heading X	Heading Y	Heading Z
0	0	0	0	0	1	0
1	1.37	1.40	-0.16	0.6438	0.7648	-0.0223
2	2.64	2.73	-0.27	0.6235	0.7817	-0.0141
3	4.39	4.20	-0.33	0.9834	0.1811	0.009
4	4.07	4.52	-0.39	0.6213	0.7834	-0.0139
5	6.34	7.20	-0.51	0.5941	0.8025	0.0555
6	8.95	10.07	-0.78	0.6411	0.7658	0.0504
7	10.12	11.40	-0.85	0.6243	0.78	0.0432
8	12.57	14.56	-1.15	0.6341	0.7712	0.0564
9	13.50	16.82	-1.36	-0.0242	0.9987	-0.0449
10	13.98	16.22	-1.19	0.9889	0.1439	-0.0373
11	13.77	16.24	-1.28	0.6168	0.7854	0.052
12	16.50	19.33	-1.54	0.6009	0.7978	-0.05
13	16.25	19.25	-1.53	0.0402	0.9986	-0.0342

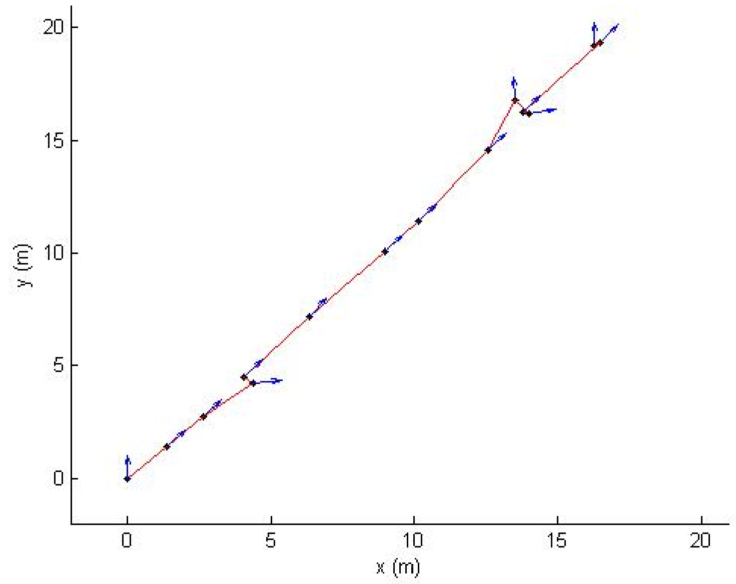


Figure 6.7 The path calculated after multiview registration.

While conducting the experiment and taking distance measurements between different poses, the terrain was assumed to be flat. However, as it can be seen in tables 6.2 and 6.3, at the end of scan registration the elevation difference between different positions are also obtained.

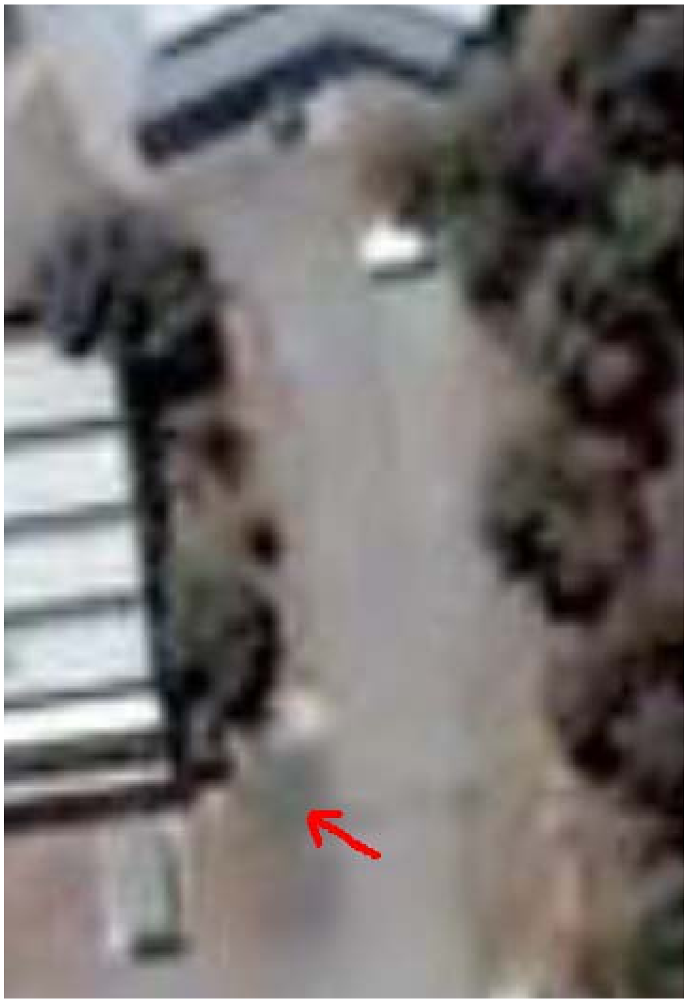


Figure 6.8 The satellite image of the site that the experiment 2 is conducted. The red arrow indicates the starting position.

CHAPTER 7

CONCLUSION

In this thesis, a solution to the autonomous navigation problem of outdoor mobile robots, using the geometric structure of the environment, is implemented. Since extension of probabilistic localization methods like Kalman filter or Monte Carlo localization algorithms to non-planar, unstructured, unbounded environments is not clear, directly using geometrical relations between 3D range images acquired at different robot poses can be the solution to the SLAM problem in outdoor.

For this purpose, a 3D range data collecting system, composing of a planar range finder and a tilt mechanism, was designed and built. Using data collected by this system at different poses, the relative motion between two different poses of the robot was found by using a variant of the ICP algorithm. During the registration process, not only the distance criterion, but also the angle criterion was used. The angle criterion used in this work eliminates the point correspondences having angles between their normals more than a preset threshold. Including this criterion in point correspondence finding step of the ICP algorithm was found out to give more accurate registrations. The output of the ICP algorithm, i.e. registered scans and the transformations which register the scans onto each other, is used to form the map of the environment and relatively localize the robot to its starting position. After the pairwise registration step, errors still remain in the global map formed and in the robot poses calculated, due to accumulation of the errors occurring in pairwise registrations. Correction these errors is done by using a global relaxation algorithm.

This algorithm diffuses the errors associated with pairwise registrations to the whole map.

7.1 Discussions and Future Work

Although the experiments to test the algorithm were conducted on almost planar terrain, the technique presented forms the 3D map of the environment and localizes the robot in 6D. So, the technique can be used in unstructured terrain, provided that some moderate estimation of the relative motion between consecutive poses is provided as an input to the algorithm. This input can be obtained by inertial measurement units, odometry or even GPS.

Instead of only using surface normals and point-to-point distances as the matching criteria, reflectivity values collected from the laser range finder can be used for point matching. This should make the registration process more robust, as normals calculated by the proposed method is not very reliable at vegetated areas in the 3D scan. Instead of using only active vision to collect data, a camera can be calibrated with the scanner (or a 3D time of flight camera can be used) in order to collect color data along with the depth values. The color difference can be a good constraint while finding corresponding points.

The registration of two point clouds takes about 5s on a Pentium-M 1700MHz Processor, and global relaxation of the map with 14 scans in experiment 2 took about 1 hour to complete. Even though the pairwise matching step takes a relatively short time and a robot can localize itself relative to the previous pose while going for the next pose, the global relaxation algorithm's performance is not satisfactory and needs to be speeded-up. A faster processor than the one which the tests are conducted on is also needed to speed up the processes.

For navigation of the robot in the environment, obstacles must be detected in the map. An algorithm that detects the obstacles and extracts the traversable terrain might be developed. Another improvement can be done on the map representation.

Currently the final map formed consists of only points. Planes or 3D shapes can be fitted to these points. This way, not only a memory efficient map representation would be achieved, but also semantic maps consisting of rocks, buildings, trees, etc. can be formed.

Maybe the most important enhancement to the implemented method would be the elimination of the need to stop for acquiring the range data. This way, a real time mapping would be possible and the robot would navigate much faster in the environment. This can be achieved by using a high-speed communication between the data collection system and the robot, and synchronizing the inertial navigation sensors and the odometry with the tilting mechanism of the data collection system.

REFERENCES

[1] Bailey, Tim, “Mobile Robot Localization and Mapping in Extensive Outdoor Environments” Ph.D. Thesis, Department of Aerospace, Mechanical and Mechatronic Engineering, University of Sydney, 2002.

[2] Borenstein, J., Everett, H.R., Feng, L., “Where am I? Sensors and Methods for Mobile Robot Positioning”, University of Michigan, April 1996.

[3] Nüchter, A., Hartmut, S., Lingemann, K., Hertzberg, J., Thrun, S., “6D SLAM with an Application in Autonomous Mine Mapping”, Robotics and Automation, IEEE International Conference on Robotics and Automation. Vol. 2, pp. 1998-2003, 2004.

[4] Georgiev, A. and Allen P.K. , *Localization Methods for a Mobile Robot in Urban Environments* IEEE Transactions on Robotics, Vol.20, No.5, October 2004, pp.851-864.

[5] Singhal, A., “Issues in Autonomous Mobile Robot Navigation”, Technical Report, University of Rochester, 1997.

[6] Spero, D.J., “A Review of Outdoor Robotics Research”, Technical Report, Monash University.

[7] Thrun, S., “Robotic Mapping: A Survey”, *Exploring Artificial Intelligence in the New Millennium*, Morgan Kaufmann Publishers Inc., San Francisco, CA, 2003.

[8] Kuipers, B., Byun Yung-Tai, “A Robot Exploration and Mapping Strategy Based on a Semantic Hierarchy of Spatial Representations”, *Toward Learning Robots*, MIT Press, Cambridge, MA, 1993.

[9] Choset, H., “Sensor Based Motion Planning: The Hierarchical Generalized Voronoi Graph” Ph. D. Thesis, California Institute of Technology, 1996.

[10] Gümrükçü, G., “Sensor Based Coverage Path Planning for a Mobile Robot”, MSc. Thesis, METU, 2003.

[11] Thrun, S., Bücken, A., “Integrating Grid-Based and Topological Maps for Mobile Robot Navigation”. In Proceedings of the 13th National Conference on Artificial Intelligence AAAI, August 1996.

[12] Thrun S., “Learning Maps for Indoor Mobile Robot Navigation”, Artificial Intelligence, Volume: 99, No. 1, pp. 27-71, 1998.

[13] Dissanayake, N. W. M. , Newman P., Durrant-Whyte, H. F. , “A Solution to the Simultaneous Localization and Map Building (SLAM) Problem”. IEEE Transactions on Robotics and Automation, Vol. 17, No. 3, pp. 229-241, June 2001.

[14] Leonard, J. J. and Durrant-Whyte, H. F. , "Simultaneous Map Building and Localization for an Autonomous Mobile Robot", IEEE / RSJ Int. Conf. on Intelligent Robots and Systems, pages 1442-1447, Osaka, Japan, 1991.

[15] Thrun, S., Burgard, W. ,Fox, D., A Probabilistic Approach to Concurrent Mapping and Localization for Mobile Robots, Machine Learning, v.31 n.1-3, p.29-53, April/May/June 1998.

[16] Fox, D., Thrun, S., Dellaert, F. and Burgard, W., “Particle Filters for Mobile Robot Localization,” In *Sequential Monte Carlo Methods in Practice,* A. Doucet, N. De Freitas, and N. Gordon, Eds. New York: Springer Verlag, 2000.

[17] Thrun, S., Fox, D., Burgard, W., Dallaert, F., “Robust Monte Carlo Localization for Mobile Robots”, Artificial Intelligence, v.128 n.1-2, p.99-141, May 2001.

[18] Kwok, C.T., Fox, D. and Meil, M., “Real-time Particle Filters”. In Advances in Neural Information Processing Systems 15, 2002.

[19] Thrun, S., Bugard, W. And Fox, D. “A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping.” In *International Conference on Robotics and Automation*, pages 321–328, 2000.

[20] Mahon, I. and Williams, S. “Three-dimensional robotic mapping.” In *Australasian Conference on Robotics and Automation*, 2003.

[21] Nüchter, A., Lingemann, K. and Hertzberg, J. “Extracting Drivable Surfaces in Outdoor 6D SLAM”, in *Proceedings of the 37nd International Symposium on Robotics (ISR '06) and 4th German Conference Robotik 2006*, ISBN 3-18-091956-6, Munich, Germany, 2006.

[22] Nüchter, A., Surmann, H., Hertzberg, J. “Automatic Classification of Objects in 3D Laser Range Scans”. In: Proc. 8th Conf. Intelligent Autonomous Systems (IAS-8), Amsterdam, The Netherlands, IOS Press, pp.963--970, 2004.

[23] Nüchter, A., Lingemann, K., Hertzberg, J., and Surmann, H. “Heuristic-Based Laser Scan Matching for Outdoor 6D SLAM”, in *KI 2005: Advances in Artificial Intelligence. 28th Annual German Conference on AI, Proceedings.* Springer (Berlin) LNAI vol. 3698, ISBN 3-540-28761-2, pages 304-319. Koblenz, Germany, September 2005.

[24] Brenneke, C., Wulf, O., Wagner, B. “Using 3D Laser Range Data for SLAM in Outdoor Environments”. IEEE/RSJ Conference on Intelligent Robots and Systems, Oct. 2003.

[25] Surmann, H., Nüchter, A., and Hertzberg, J. “An autonomous mobile robot with a 3D laser rangefinder for 3D exploration and digitalization of indoor environments.” Journal Robotics and Autonomous Systems, 45(3- 4):181 - 198, December 2003.

[26] Wulf, O. and Wagner, B.: “Fast 3D-Scanning Methods for Laser Measurement Systems.” 14th International Conference on Control Systems and Computer Science (CSCS14), July 2-5, 2003, Bucharest, Romania.

[27] Howard, A., Wolf, D.F., and Sukhatme, G.S., "Towards 3D Mapping in Large Urban Environments," In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 419-424, Sendai, Japan, Sep 2004.

[28] Wolf, D.F., Howard, A., and Sukhatme, G.S., "Towards Geometric 3D Mapping of Outdoor Environments Using Mobile Robots," In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1258-1263, Aug 2005.

[29] Pfaff, P., Triebel, R., Burgard, W., **"**An Efficient Extension to Elevation Maps for Outdoor Terrain Mapping and Loop Closing**"**. International Journal of Robotics Research, Volume 26 , Issue 2, pp. 217-230, 2007.

[30] Hähnel, D., Schulz, D., Burgard, W., "Map Building with Mobile Robots in Populated Environments." In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2002.

[31] Huber, D.F. and Hebert, M., "A New Approach to 3-D Terrain Mapping". *Proceedings of the 1999 IEEE/RSJ International Conference on Intelligent Robotics and Systems (IROS '99)*, IEEE, October, 1999, pp. 1121-1127.

[32] Johnson, A.E. , "Spin-Images: A Representation for 3-D Surface Matching", PhD Thesis, Carnegie Mellon University, 1997.

[33] SICK LMS Technical Description

[34] SICK LMS Telegram Listing

[35] Besl, P.J., McKay N.D., "A Method for Registration of 3-D Shapes.", IEEE Trans. Pattern Anal. Mach. Intell. 14(2): 239-256 (1992).

[36] Zhang, Z., "Iterative point matching for registration of free-form curves and surfaces.", International Journal of Computer Vision, v.13 n.2, p.119-152, Oct. 1994.

[37] Chen, Y. and Medioni, G. "Object Modeling by Registration of Multiple Range Images," *Proc. IEEE Conf. on Robotics and Automation*, 1991.

[38] Lorusso, A., Eggert, D. W., Fisher, R. B., "A comparison of four algorithms for estimating 3-D rigid transformations.", Proceedings of the 1995 British conference on Machine vision (Vol. 1), p.237-246, July 1995, Birmingham, United Kingdom.

[39] Cang Ye, Borenstein, J., “Characterization of a 2-D Laser Scanner for Mobile Robot Obstacle Negotiation.”, ICRA 2002: 2512-2518.

[40] Surmann, H., Nüchter, A., Lingemann, K. and Hertzberg, J., “6D SLAM — Prelımınary Report on Closıng the Loop in Sıx Dımensıons.” *Fraunhofer Institute for Autonomous Intelligent Systems.*

[41] Rusinkiewicz, S and Levoy, M, “Efficient Variants of the ICP Algorithm.” Third International Conference on 3D Digital Imaging and Modeling (3DIM). June 2001.

[42] Bentley, J.L., “Multidimensional binary search trees used for associative searching.”, Communications of the ACM, v.18 n.9, p.509-517, Sept. 1975 .

[43] Wikipedia, Kd-tree, http://en.wikipedia.org/wiki/Kd_tree, October 2007.

[44] Pulli, K. *Surface Reconstruction and Display from Range and Color Data*, Ph.D. Dissertation, University of Washington, 1997.

[45] Weik, S. “Registration of 3-D Partial Surface Models Using Luminance and Depth Information,” *Proc. 3DIM*, 1997.

[46] Turk, G. and Levoy, M. “Zippered Polygon Meshes from Range Images,” *Proc. SIGGRAPH*, 1994.

[47] Masuda, T., Sakaue, K., and Yokoya, N. “Registration and Integration of Multiple Range Images for 3-D Model Construction,” *Proc. CVPR*, 1996.

[48] Arun, K. S., Huang T. S., Blostein S. D., “Least-Squares Fitting of Two 3-D Point Sets.”, IEEE Transactions on Pattern Analysis and Machine Intelligence, 9,5, 1987, pp. 698-700.

[49] Horn B. K. P., Hilden H. M., Negahdaripour S., “Closed-form Solution of Absolute Orientation Using Orthonormal Matrices,” Journal of the Optical Society of America, Series A, 5, 7, 1988, pp. 1127-1135.

[50] Horn, B. K. P., "Closed-form Solution of Absolute Orientation Using Unit Quaternions," Journal of the Optical Society of America, Series A, 4, 4, 1987, pp. 629-642.

[51] Walker M. W., Shao L. and Volz R. A., "Estimating 3-D Location Parameters Using Dual Number Quaternions," CVGIP: Image Understanding, 54, 3, 1991, pp. 358-367.

[52] Bentley, J.L., "K-d trees for semidynamic point sets.", Proceedings of the sixth annual symposium on Computational geometry, p.187-197, June 07-09, 1990, Berkley, California, United States.

[53] Borenstein, J. and Feng, L., "Correction of Systematic Odometry Errors in Mobile Robots". Proceedings of the 1995 International Conference on Intelligent Robots and Systems (IROS '95) pp. 569-574, 1995.

[54] Yarkınoğlu, O., "Computer Aided Manufacturing (CAM) Data Generation for Solid Freeform Fabrication", M.Sc. Thesis, METU, 2007.

[55] Nishino, K. and Ikeuchi, K., " Robust Simultaneous Registration of Multiple Range Images". The 5th Asian Conference on Computer Vision, January 2002, Melbourne, Australia.

[56] Pulli, K., "Multiview Registration for Large Data Sets". Second International Conference on 3D Digital Imaging and Modeling (3DIM'99), pp. 160-168, Ottawa, Canada, October 1999.

[57] Silva, L., Bellon, O. R. P., Boyer, K.L., "Range Image Registration Using Enhanced Genetic Algorithms". International Conference on Image Processing, Barcelona, Spain, September 2003.

[58] Matabosch, C., Salvi, J., Fofi, D., and Meriaudeau, F., "Range image registration for industrial inspection". Machine Vision Applications in Industrial Inspection XIII, MVAII05, San Jose, California, USA, January16–20, 2005.

[59] Lu, F., and Milios, E., "Globally Consistent Range Scan Alignment for Environmental Mapping". Autonomous Robots Volume 4, Issue 4, pp.333 - 349 ,October 1997.

[60] Bergevin, R., Soucy, M., Gagnon, H. and Laurendeau, D., “Towards a General Multi-View Registration Technique”. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 18, No 5, pp.540-547, May 1996.